זיהוי מחלות אצל צמחים

בית ספר: תיכון בן גוריון - נס ציונה
התמחות: לימוד מכונה
תעודת זהות: 328494836
שם: כפיר איתן

תוכן עניינים

מבוא:

החקלאות המודרנית היא אינטנסיבית, ומושקעים מאמצים, אמצעים ומשאבים רבים כדי לגדל כמה שיותר גידולים חקלאיים ביעילות גדולה ובשטח מצומצם. מדי שנה גורמים חיידקים, פטריות, טפילים וחרקים נזקים בהיקף של מיליוני שקלים לחקלאות הישראלית. התפשטות גורם מחלה ללא טיפול מתאים יכול לגרום לקריסתו של ענף ולקריסה כלכלית של העוסקים בו.
אחת הסיבות להתפשטות מהירה של מחלות בבעלי חיים ובצמחים היא צפיפות. שדות צפופים של חיטה, חממות, מטעים עם אלפי צמחים וגם לולים, דירים ורפתות שמאוכלסים במאות ולעתים אלפים של בעלי חיים.
מחלות צמחים לא גורמות רק לפגיעה כלכלית, בעבר היו מקרים שבהם מגפה בצמחים גרמה לרעב המוני, מוות ואף לשינוי ההיסטוריה האנושית(לדוגמא - מחלת הכימשון שתקפה את תפוחי האדמה באירלנד במאה ה- 19 והביאה למותם של כמיליון בני אדם ברעב ומחלות ולהגירה גדולה לארה"ב).

מטרת הפרויקט היא לעזור לחקלאים לזהות מהר יותר את העצים החולים וכך לטפל מהר יותר בהם. רחפן עם מצלמה אשר יעבור בשדות חקלאים, יצלם ויסרוק את הצמחים ובמידה והוא זיהה צמח חולה, הוא ישלח הודעה לחקלאי וכך הוא ידע על מצב הצמחים שלו בלי להסתובב בשטח.

בחרתי בנושא זה מכיוון שאני מאוד אוהב צמחים, עולם החקלאות מאוד מעניין אותי ובעתיד הייתי רוצה לעזור ולפתח את עולם החקלאות בהיבט הטכנולוגי.

במסגרת פרויקט זה הפתרון המוצע מאפשר לנתח את מצב הצמחים ללא בני אדם בשטח. הקוד מסווג בין שלושה מצבים:

1. צמח בריא.
2. חילדון (RUST) - פטריה אשר מתפתחת על עלים שרטובים למשך מספר שעות בעקבות השקיה, גשם וטל. נבגי הפטרייה הינם בצבע צהוב עד חום כהה ונראים כמו חלודה ומכאן שמה. הפטרייה פעילה בין מרץ לספטמבר כאשר בטמפרטורות נמוכות הפטרייה מתפשטת ומתפתחת ובטמפרטורות גבוהות נעלמת.
3. קימחון (POWDERY) - משפחה רחבה של פטריות נפוצות אשר פוגעות בצמחים רבים, סימני הימצאותן בולטים מאוד ונראים כמו תפטיר קמחי אשר מופיע על העלים התחתונים וללא טיפול, ילכו ויתפשטו עד כדי כיסויו המוחלט של הצמח. המחלה מתפרצת בעיקר בעונות המעבר מעצם היותה חובבת טמפרטורות מתונות (סביב 23 מעלות) ולחות גבוהה, הקימחון נפוצה גם בקיץ ולעיתים שורדת היטב גם בלחות נמוכה.

<u>אתגרים מרכזיים:</u>

אחת הבעיות במרכזיות שיכולות להשפיע לי על סיווג המחלות היא הכמות הלא מאוזנת של תמונות בקטגוריות השונות אשר יכולה להוביל לנטייה של המחשב למחלה מסוימת בשל ההפרש בין כמות התמונות.
בעיה נוספת בפרויקט היא גודלי התמונות, מבחינת פיקסלים, התמונות הן 3000*4000 פיקסלים, הגודל יותר מידי גדול ומעל מה שצריך.

<u>מאמרים:</u>

מאמר 1 – "מזיקים ומחלות בצמחים – זיהוי וטיפול"

מאמר 2 – "קמחון – מהם מאפייני המחלה וכיצד ניתן להדבירה?"

מאמר 3 – "חילדון"

שלב איסוף הכנה וניתוח הנתונים:

דרך איסוף המידע

כדי לאמן מודל למידה עמוקה יש צורך בכמות נכבדה של תמונות מסווגות מראש. באימון מלא נדרשות עשרות אלפי ואפילו מאות אלפי תמונות כדי להגיע לאימון המפיק דיוק טוב.

את מאגר התמונות המסווגות ניתן ליצור במגוון דרכים.

1. ייצור עצמי של התמונות המסווגות על ידי צילום צמחים ומיונן לפי מחלות. שיטה זו נחשבת לאמינה ולמדויקת מכולם אך היקרה וצרכנית הזמן הגדולה מכולן. גישה זו מתאימה לחברות ענק כגון amazon.
2. חיפוש תמונות באינטרנט באמצעות מנוע חיפוש. גם שיטה זו אורכת זמן רב ואמינותה מוטלת בספק בשל סיווגים שגויים.
3. שימוש במאגר מידע קיים.

אני בחרתי בגישה השלישית והורדתי מאגר מידע מהאינטרנט.
מבנה הנתונים שממנו לקחתי את הנתונים הוא Plant disease recognition dataset אשר נוצר על ידי RASHIK RAHMAN בשנת2021 והועלה לאתר KAGGLE, אתר המאפשר לחקור, לבנות ולהוריד מודלים שונים אשר מבוססים על מידע מהאינטרנט ומחקרים.

תיקיית ה- TRAIN מכילה 3 תיקיות שונות אשר כל תיקייה מציגה דוגמאות שונות של מחלות שונות אצל הצמחים. סך כל התמונות בתיקיית הTRAIN הוא 1322.

תיקיית ה- TEST גם מכילה 3 תיקיות שונות אשר בתוכן יש 150 תמונות שונות. מספר זה של תמונות בעבור סיווגים רבים כל כך נחשב יחסית קטן, אולם השימוש בשיטת CNN מאפשר הגעה לדיוק מקסימלי באמצעות כמות זו של תמונות.

תיאור וניתוח הנתונים הגולמיים

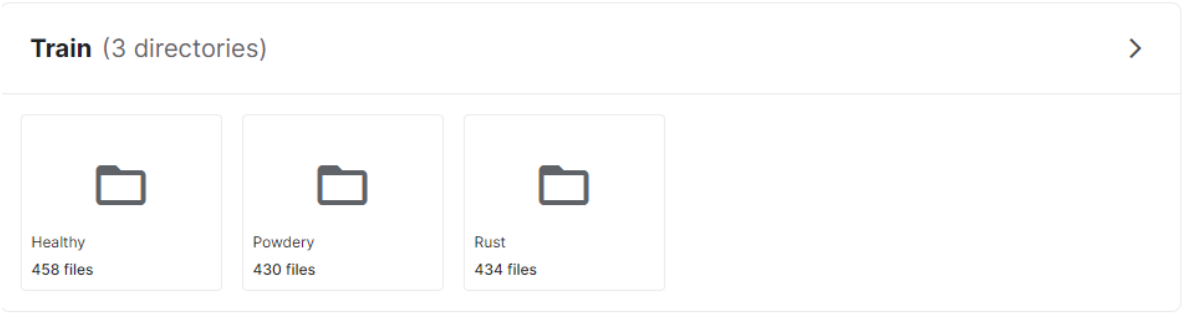
הנתונים נוצרו באמצעות תמונות של עלים בשטח פתוח. התמונות כוללות שלושה מצבים של העלים- בריאים, חולים בחילדון וחולים בקמחון.

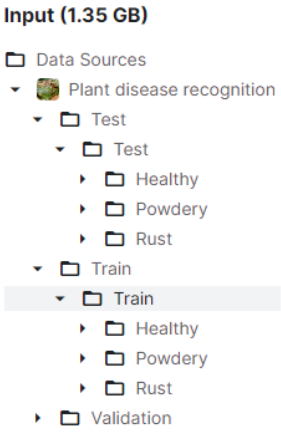
התמונות הן 4000*3000 פיקסלים אך בעיבוד התמונות אעביר אותן לממד קטן יותר.

התמונות נרשמו אוטומטית כך שהעלים פחות או יותר ממורכזות ותופסות בערך את אותו מקום בכל תמונה.

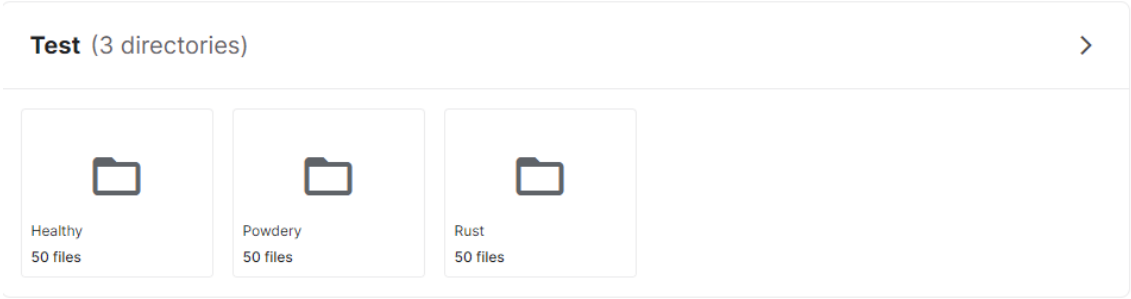
דוגמא לתמונות ממאגר הנתונים:

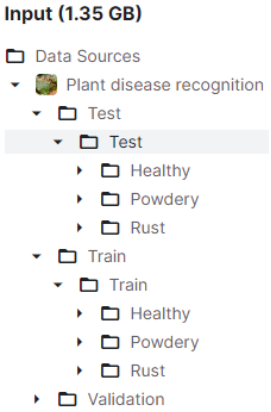
תיקיית "Train" המכילה את הנתונים לאימון המודל:





תיקיית "Test" המכילה את הנתונים לבחינת המודל:





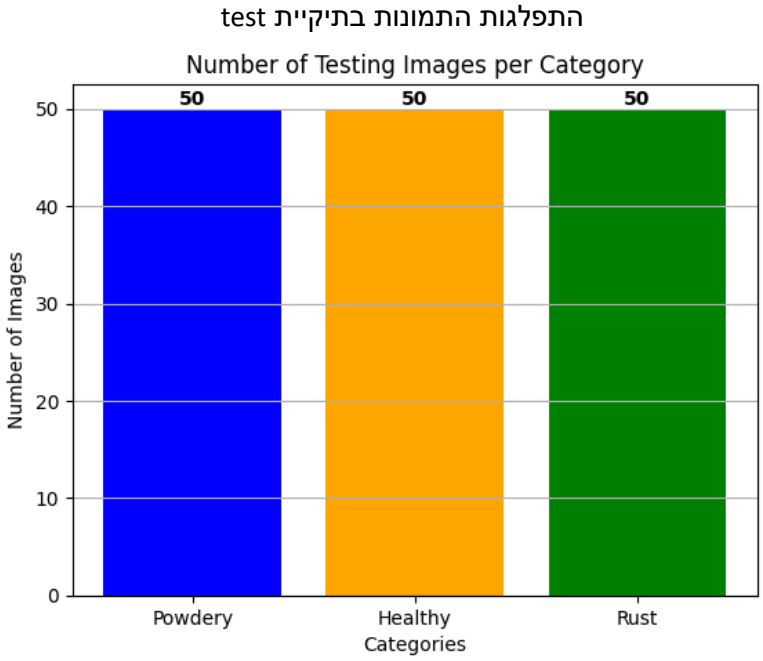
ניתן לראות כי כל תיקייה מכילה מצב אחר של הצמח:

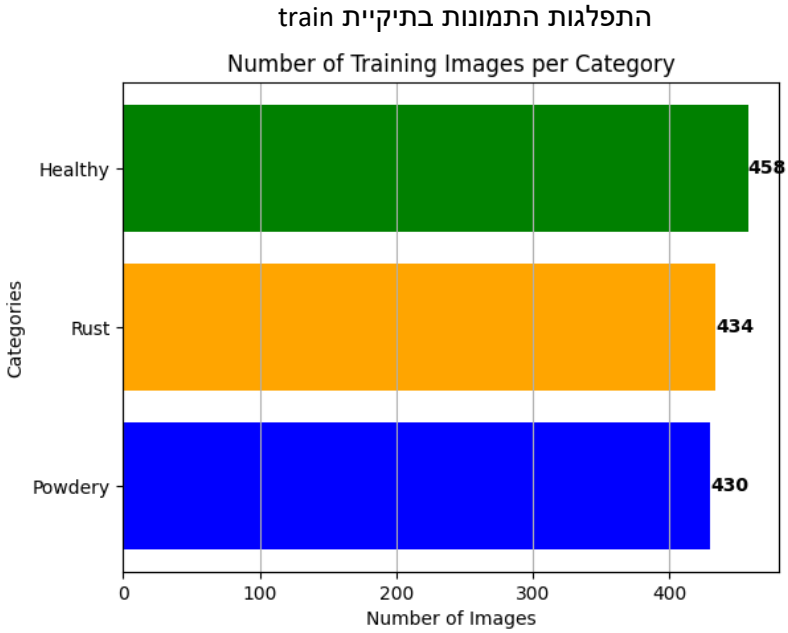
Healthy – בריא.

Powdery – קמחון.

Rust – חילדון.

התפלגות התמונות בDATA





האתגרים השונים שיש ב - Data Set

בלימוד מכונה אנו נדרשים ללמד את המכונה לסווג, לעשות משהו כמו בני אדם. בנושא שאני בחרתי, אני צריך ללמד את המכונה לעשות משהו שלא כל בני האדם יודעים לעשות – לסווג בין מצבים רפואיים וסוגי מחלות של צמחים על ידי העלים שלהם, שזה דבר שהרבה פעמים רק אנשים שעובדים בתחום יודעים לעשות.

אתגר נוסף הוא שמאגר הנתונים יחסית מאגר לא כה קל ללמידה ואימון המודל, הוא כולל בתוכו מספר רב של תמונות עם חלקים חלקיים מהעלים ויש חשיבות רבה לפרטים קטנים. נוסף על כך, יש גם מספר תמונות שהרקע תופס חלק גדול מהתמונה, דבר שסביר שנבע מאיסוף נתונים ברמה בינונית ובאופן שלא לוקח בחשבון את המפתח.

פתרון – המודל צריך להיות יציב מאוד ובנוי מספיק טוב על מנת להצליח ללמוד גם עם הנתונים האלו, אם הדבר יעשה בהצלחה המודל ישתפר בצורה גדולה יותר ממה שהיה משתפר באימון שנעשה עם דוגמאות קלות יותר.

זאת ועוד, גודלי התמונות, מבחינת פיקסלים, מאוד גדולות ומעל מה שצריך. זה משפיע על יכולת הלמידה של המודל ועל זמן לימודו.

פתרון – בעיבוד התמונות אעביר אותן לממד קטן יותר.

שלב בנייה ואימון המודל:

בחירת השיטה

בפניי עמדו מספר שיטות אשר יכולתי לממש אותן בפרויקט שלי.

RNN(Recurrent Neural Network) – זוהי שיטה דינמית אשר מתבססת על רשת נוירונים.
החיסרון העיקרי שלה זה שהיא איטית ומסורבלת מידי, מנגד, היתרון העיקרי שלה זה שהיא מתאימה לעיבוד נתונים עצומים ויכולה לזכור אותם לאורך זמן רב לאחר אימון המודל. לא בחרתי בשיטה זו מכיוון שהיא מתאימה יותר לזיהוי קול ולכתיבת טקסט והפרויקט שלי מתעסק בזיהוי בתמונות.

ANN(Artificial Neural Network) – היא השיטה הפשטנית ביותר של רשת עצבית מלאכותית.
החיסרון העיקרי שלה זה שהיא לא מתאימה למצבים מורכבים מידי, כמו עיבוד עם תמונות ולכן לא בחרתי בשיטה זו.

לאחר שעברתי על מספר שיטות, בחרתי בשיטת **CNN(Convolutional Neural Network)** ליישום הפרויקט שלי.
שיטה זו היא היעילה ביותר בלימוד מכונה. היא מתבססת על רשת נוירונים בתהליך קונבולוציוני (משתמשת במקום gradiant desent בכפל מטריצות, כלומר, במקום לנסות למצוא את השיפוע, המינימום האפשרי נמצע על ידי כפל מטריצות). שיטה זו תוכננה במיוחד לעיבוד נתוני פיקסלים ומשתמשים בה על מנת לזהות ולעבד תמונות.
בחרתי בשיטה זו מכיוון שאני מתעסק בזיהוי ובעיבוד תמונות וזוהי השיטה היעילה ביותר למטרה זו.

בחירת המודל

ליישום הפרויקט בחרתי להשתמש במודל **Sequential**. מודל זה הוא ערימה לינארית של שכבות. זה מאפשר ליצור רשת עצבית על ידי ערמת שכבות זו על גבי זו באופן רציף, מקלט לפלט. כל שכבה במודל מחוברת לשכבה הקודמת והאחרת.

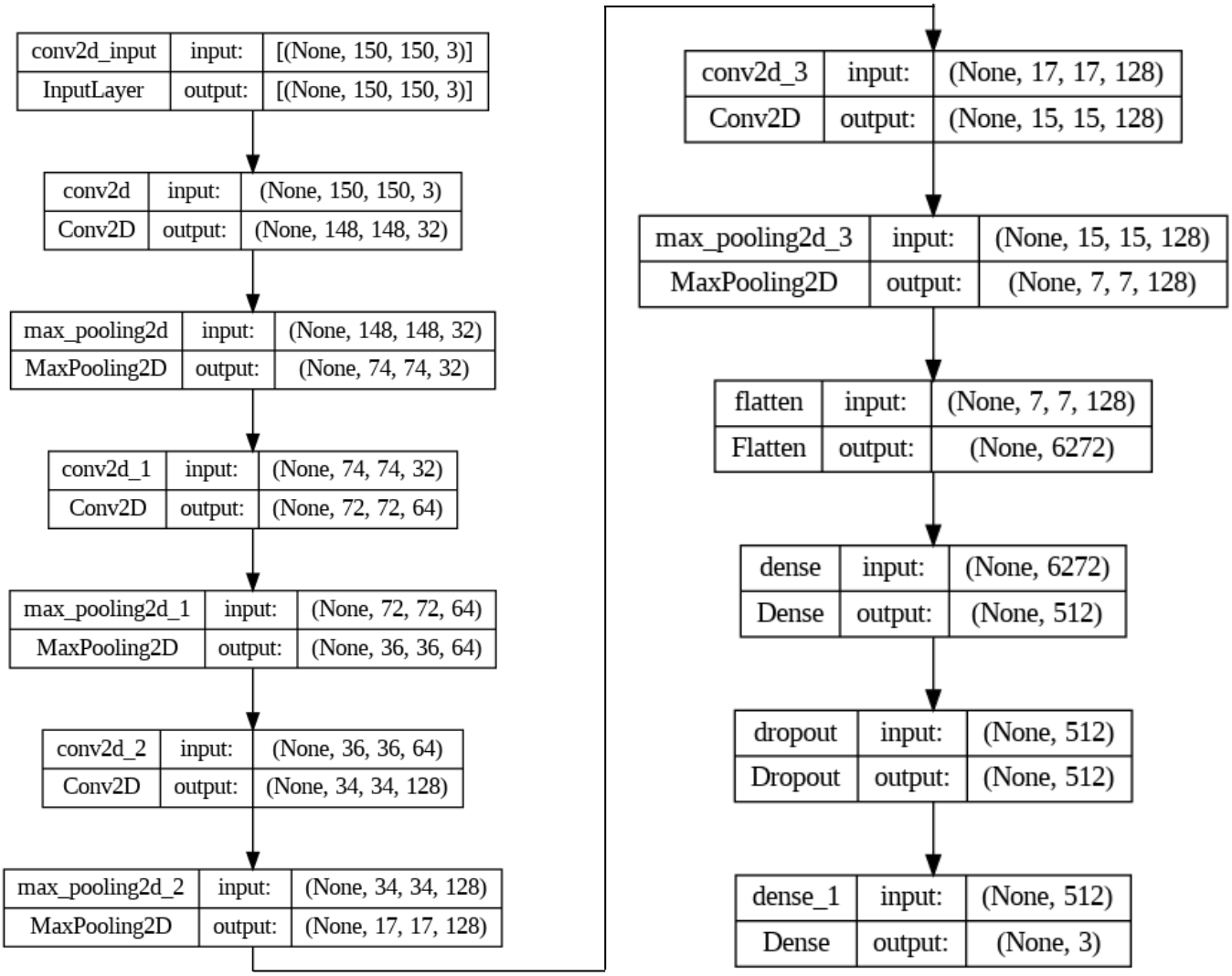
המודל **Sequential** קל להבנה ולשימוש, במיוחד למתחילים. זה מספק דרך פשוטה לבנות רשת עצבית פשוט על ידי הוספת שכבות ברצף(בעזרת הפעולה ()add על המודל). המודל תומך באימון יעיל באמצעות אלגוריתמי אופטימיזציה פופולריים ופונקציות אובדן. אפשר בקלות להרכיב את המודל עם פונקציית האופטימיזציה וההפסד הרצויה. בנוסף, הוא מספק תמיכה מובנית למדדים להערכת הביצועים של המודל במהלך האימון.
המודל **Sequential** מציע גמישות בעיצוב מודלים של למידה עמוקה. הוא נותן את האפשרות להוסיף סוגים שונים של שכבות, כגון שכבות קונבולוציוניות, שכבות pooling, שכבות dense, שכבות Dropout ועוד. זה מאפשר לך לבנות מגוון רחב של ארכיטקטורות עבור משימות שונות.

לסיכום, המודל **Sequential** מציע דרך פשוטה ויעילה לבנות רשתות עצביות, מה שהופך אותו לבחירה פופולרית למשימות למידה עמוקה רבות. היתרונות שלו טמונים בפשטות, קלות היישום, הגמישות, האימון היעיל והתאימות עם פונקציות אחרות של Keras. לכן בחרתי במודל זה ליישום הפרויקט.

תיאור גרפי של המודל שעליו בוצע האימון

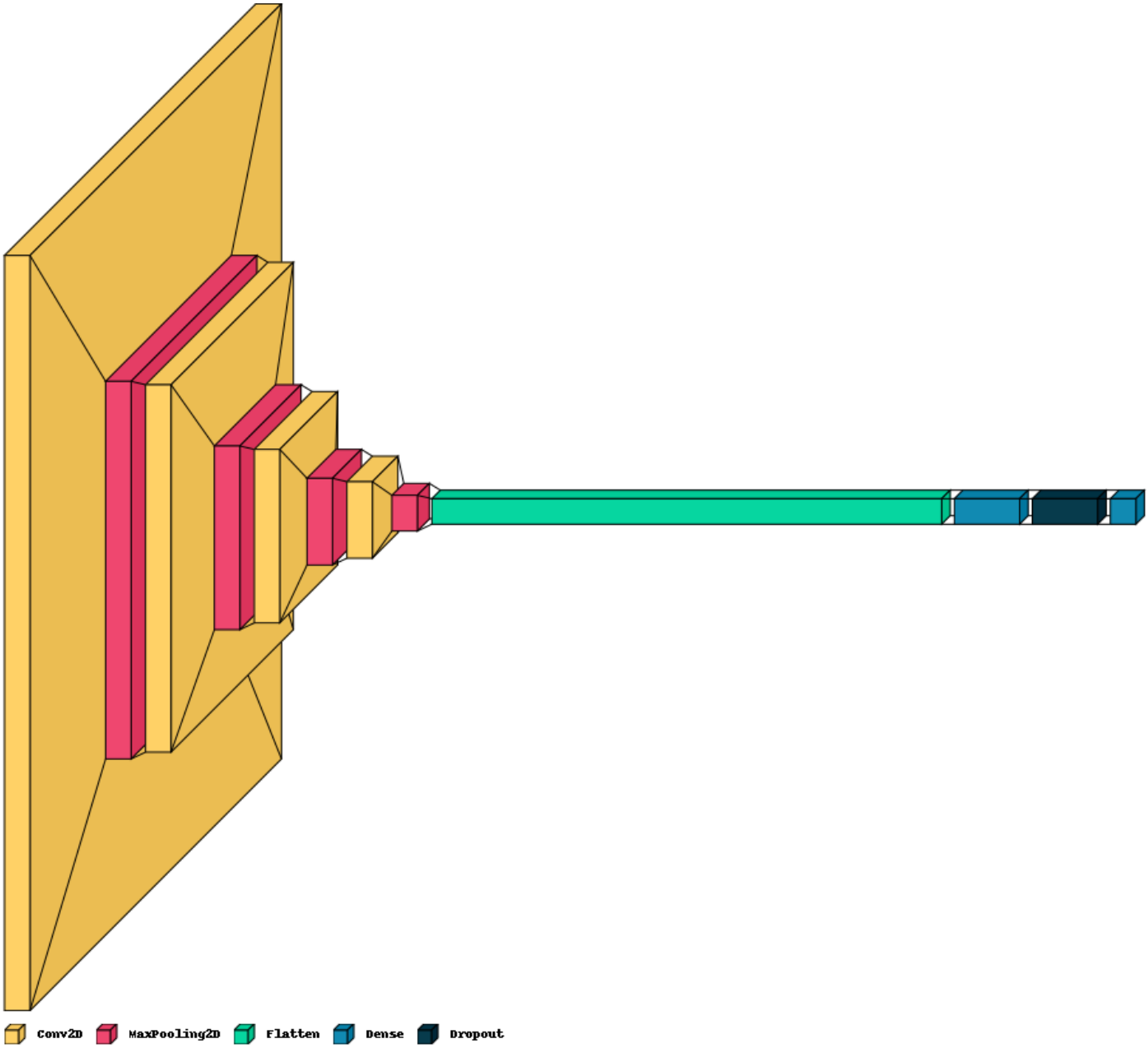
בניית המודל –

```
# Create the model
model = Sequential()

# Add convolutional layers
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten()) # Flatten the output tensor from the
convolutional layers

# Add dense layers for classification
model.add(Dense(units=512, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(units=3, activation='softmax'))
```



```
!pip install visualkeras
import visualkeras
from IPython.display import display
display(visualkeras.layered_view(model, legend=True))
```

האלגוריתם הזה מספק הדמיה של הארכיטקטורה של המודל. היא יוצרת דיאגרמה שמראה כל שכבה של המודל זו על גבי זו, כך שלכל סוג שכבה יש צבע שונה.
ההדמיה נראית כך:



הסבר על סוגי השכבות ברשת

1. Conv2D – שכבה זו היא אבן הליבה של רשת CNN, אשר נושאת את החלק העיקרי בהליך החישוב. מה שלמעשה מבצעת שכבה זו זה כפל מטריצות בין מטריצה ריבועית אחת של פרמטרים ניתנים לאימון (מטריצה זו היא למעשה פילטר בשכבה, גודלה הוא כגודל ה kernel size) לבין כל המטריצות הריבועיות השונות, באותו גודל של הkernel size, אשר קיימות בתוך המטריצה שנקלטה אל השכבה. הפלט כתוצאה מהליך כפל המטריצות, הוא מטריצה ריבועית שנקראת Feature map כי היא מחלצת עבורנו את המאפיינים הרלוונטיים מהקלט. אותה Feature map תועבר כקלט לשכבות הבאות.
2. MaxPooling2D – אלה מבצעות איגום (קליטה ופליטה של נתונים במערכת מחשב תוך שימוש במאגר זמני חיצוני) מקסימלי על n*n בלוקים. איגום מקסימלי הוא תהליך שבו הקלט מחולק לאזורים כאשר מכל אזור נלקח הערך המקסימלי שבו. כך מתאפשר להקטין את הגודל של הקלט ולשמר רק את הערכים החשובים והמשמעותיים. שכבה זו עוזרת לחלץ תכונות דומיננטיות תוך הפחתת המורכבות החישובית.
3. Flatten – מטרתה להמיר את הפלט מהשכבות הקונבולוציוניות לווקטור חד ממדי שיוכל להתקבל כקלט בשכבת הDense.
4. Dense – שכבות אשר ממוקמות בדרך כלל לפני שכבת הפלט ויוצרות את השכבות האחרונות של ארכיטקטורת הCNN. כל נוירון בשכבה זו מקבל קלטים מכל הנוירונים בשכבה הקודמת, מבצע קומבינציה לינארית שלהם על ידי משקלם, מוסיף את מקדם הbias, ומפעיל את פונקציית האקטיבציה כדי לקבל פלט. כך הן למעשה מקשרות בין כל קלט שנכנס לשכבה לכל פלט שיוצא ממנה.
5. Dropout – שכבת Dropout מוסיפה אי וודאות למערך הפלט של השכבה הקודמת. שכבה זו עוזרת למנוע התאמת יתר(הסבר נרחב תחת הכותרת רגולציה).

השכבות במודל

```
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(150, 150, 3)))
```

שכבת הקלט, Conv2D – זוהי שכבה קונבולוציונית עם 32 מסננים. לכל מסנן גודל של 3*3 פיקסלים. פונקציית ההפעלה המשמשת היא ReLU, המציגה אי-לינאריות למודל. צורת הקלט היא (150, 150, 3), המציינת את ממדי התמונה (גובה, רוחב, ערוצים). כאשר התמונות קלט בגודל 150*150 עם 3 ערוצי צבע (RGB).

```
model.add(MaxPooling2D((2, 2)))
```

שכבת MaxPooling2D המשתמשת בחלון של 2*2.

```
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
```

שכבת Conv2D השנייה, יש לה 64 מסננים בגודל 3*3 עם פונקציית הפעלה ReLU.

```
model.add(MaxPooling2D((2, 2)))
```

שכבת MaxPooling2D המשתמשת בחלון של 2*2.

```
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
```

שכבת Conv2D השלישית, יש לה 128 מסננים בגודל 3*3 עם פונקציית הפעלה ReLU.

```
model.add(MaxPooling2D((2, 2)))
```

שכבת MaxPooling2D המשתמשת בחלון של 2*2.

```
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
```

שכבת Conv2D הרביעית, יש לה 128 מסננים בגודל 3*3 עם פונקציית הפעלה ReLU.

```
model.add(MaxPooling2D((2, 2)))
```

שכבת MaxPooling2D המשתמשת בחלון של 2*2.

```
model.add(Flatten())
```

שכבה המשטחת את הפלט הרב-ממדי מהשכבות הקונבולוציוניות הקודמות לווקטור בודד.

```
model.add(Dense(units=512, activation='relu'))
```

השכבה הצפופה (Dense) הראשונה, יש בה 512 נוירונים ו פונקציית הפעלה ReLU.

```
model.add(Dropout(0.5))
```

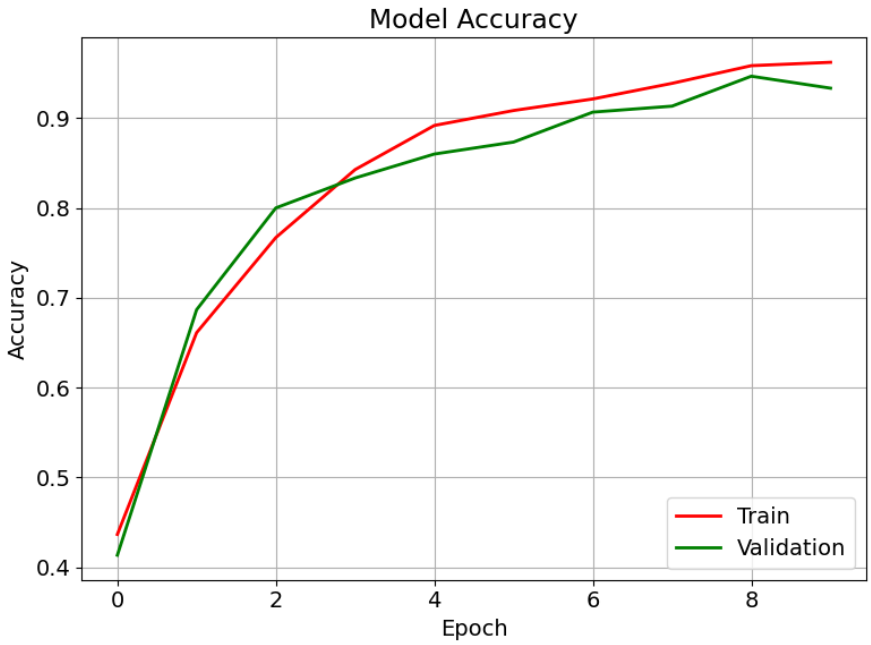
שכבת Dropout מפילה באופן אקראי 0.5 מהנוירונים במהלך האימון כדי לשפר את לימוד המודל.

```
model.add(Dense(units=3, activation='softmax'))
```

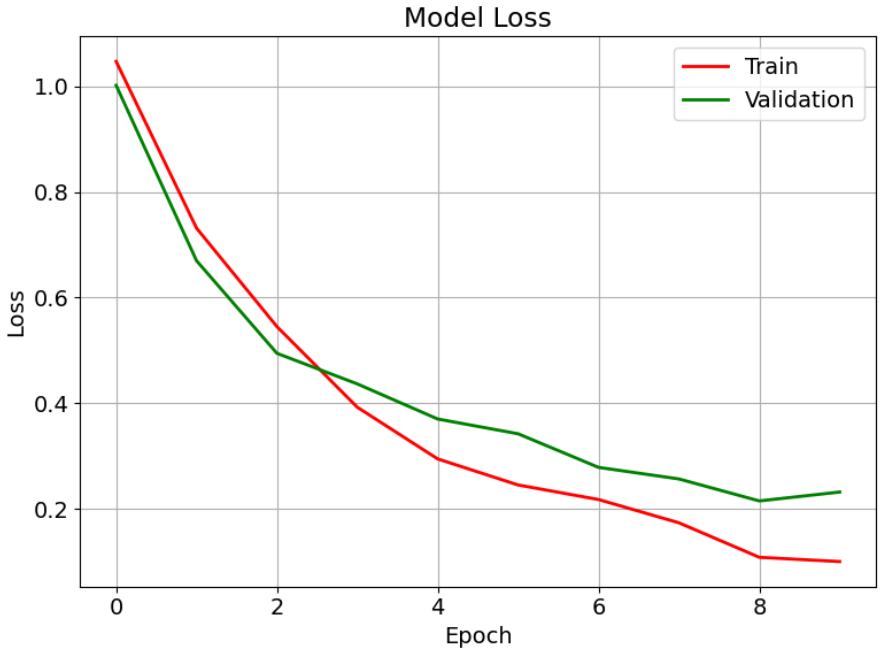
השכבה הצפופה האחרונה, יש בה 3 נוירונים (כמספר קטגוריות הסיווג) עם פונקציית הפעלה Softmax.

גרפים תוצאות שלב האימון

גרף המתאר את דיוק המודל לאורך שלבי האימון.
ציר X מייצג את שלבי אימון המודל (epochs).
ציר Y מייצג את ערכי הדיוק (Accuracy).
הקו האדום מייצג את הדיוק באימון של המודל. הוא מציג כיצד הדיוק באימון משתנה בכל אפוק שהמודל עובר דרכו.
הקו הירוק מייצג את הדיוק באימות של המודל. הוא מציג כיצד הדיוק באימות משתנה בכל אפוק שהמודל עובר דרכו.



גרף המתאר את השגיאה של המודל לאורך שלבי האימון.
ציר X מייצג את שלבי אימון המודל (epochs).
ציר Y מייצג את ערכי ההפסד (Loss). הפסד הוא מדד לאי התאמה בין התפוקות החזויות של המודל לבין התפוקות הרצויות. מטרת האימון היא למזער את ההפסד ולהשיג ערכים נמוכים יותר.
הקו אדום מייצג את ההפסד במהלך האימון. זה מראה כיצד ההפסד משתנה ככל שהמודל עובר בכל אפוק.
הקו ירוק מייצג את אובדן האימות. זה מראה כיצד ההפסד בערכת האימות משתנה ככל שהמודל עובר בכל אפוק.



Hyper Parameters בקוד

מודלים של למידת מכונה אינם אינטליגנטים מספיק כדי לדעת אילו היפרפרמטרים יובילו לדיוק הגבוה ביותר האפשרי במערך הנתונים הנתון. עם זאת, ערכי היפרפרמטרים כשהם מוגדרים נכון יכולים לבנות מודלים מדויקים ביותר, ולכן אנו מאפשרים למודלים שלנו לנסות שילובים שונים של פרמטרים שונים במהלך תהליך האימון ולבצע חיזויים עם השילוב הטוב ביותר של ערכי היפרפרמטרים.

הפרמטרים בקוד –

- **'Batch_size'** – מוגדר ל48 ב 'training_set' ו 'test_set'. הוא קובע את מספר הדוגמאות שמועברות דרך הרשת בכל אפוק.
- **'epochs'** – מוגדר ל10 בפונקציית 'model.fit()'. הוא קובע את מספר הפעמים בהן נעבור על כל קבוצת הנתונים לאורך תהליך האימון.
- **'learning_rate'** – מוגדר ל0.001 באופטימייזר. הוא קובע את גודל הצעד בכל תיקון המשקלים של המודל.
- **'dropout_rate '** – מוגדר ל 0.5 בשכבת 'Dropout'. זה מציין את השיעור של יחידות הקלט שיש להשליך במהלך האימון וזה עוזר למנוע התאמת יתר.
- **'filters'** – מספר המסננים שיש בכל שכבה קונבולוציונית. במודל יש 4 שכבות כאלה עם 32,64,128 מסננים.
- **'activation'** – פונקציית ההפעלה הנמצאת בשכבות הקונבולוציוניות היא ReLU ובשכבת הסיווג הסופית (השכבה הצפופה) הפונקציה היא softmax.
- **'karnel_size'** – הגודל של המסנן הקונבולוציוני. במודל זה הוא (3,3), בכל השכבות הקונבולוציוניות.
- **'loss'** – פונקציית ההפסד המשמשת לאימון המודל. בפרויקט שלי השתמשתי בפונקציית "Categorical cross-entropy ".
- **'input_shape'** – הצורה של תמונת הקלט. המודל מצפה לתמונות קלט בגודל (150,150) עם 3 ערוצים (RGB).
- **'units'** – קובע את המורכבות של השכבה ומשפיע על יכולת הלמידה של המודל. בשכבת Dense הראשונה הוא מוגדר ל512 ובשכבה השנייה שהיא גם שכבת הפלט הוא מוגדר ל-3.

אופטימיזציה

אופטימיזציה היא תהליך המתמקד בחיפוש ובשיפור המודל בקרב מטרות ספציפיות. מטרת תהליך זה היא למצוא את הפרמטרים המיטביים למודל בעבודתו עם נתוני האימון, כך שהמודל יכול לספק תוצאות טובות יותר במהלך הסיווג.

תוך כדי אימון אופטימיזציה המודל, המשקולות משתנים על מנת לצמצם את פונקציית ההפסד. אופטימיזר הוא פונקציה או אלגוריתם שמתאים את התכונות של הרשת העצבית, כגון משקלים וקצבי למידה ולפיכך, זה עוזר להפחית את ההפסד הכולל ולשפר את הדיוק.

Gradient descent – אלגוריתם אופטימיזציה המשמש כדי למזער את פונקציית האובדן של המודל. אלגוריתם מתחיל באתחול הפרמטרים של המודל באופן אקראי. לאחר מכן הוא מחשב את השיפוע (גרדיאנט) של פונקציית ההפסד ביחד לפרמטרים האלה. השיפוע מייצג את כיוון העלייה התלולה ביותר, ומציין כיצד פונקציית האובדן משתנה עם שינויים קטנים בפרמטרים. בכדי למזער את פונקציית האובדן, האלגוריתם צועד בכיוון ההפוך של השיפוע, ונע לעבר המינימום של הפונקציה. גודלו של כל שלב נקבע לפי קצב הלמידה (learning_rate), היפרפרמטר השולט בגודל עדכוני הפרמטרים.

ליישום הפרויקט שלי בחרתי להשתמש באלגוריתם האופטימיזציה Adam.
Adam (adaptive Moment Estimation) הוא אלגוריתם אופטימיזציה המשמש לעדכון המשקלים של רשת עצבית במהלך האימון. האלגוריתם מתאים את קצב הלמידה לכל פרמטר ברשת על סמך השלבים הקודמים, והוא נוטה להתכנס מהר יותר מאלגוריתמי אופטימיזציה אחרים, במיוחד עבור מערכי נתונים גדולים או מודלים מורכבים.

בחרתי להשתמש באלגוריתם Adam בשל קצב הלמידה האדפטיבי שלו כך שזה עוזר להשיג התכנסות מהירה יותר וביצועי אופטימיזציה משופרים. השימוש היעיל באחסון, החוסן לנתונים רועשים ויכולת האופטימיזציה שלו, עוזרים לאימון יעיל של המודל עבור משימות סיווג וזיהוי.

Learning_rate – בחירת קצב הלמידה תלויה במספר גורמים ויכולה להשפיע באופן משמעותי על הביצועים של הרשת במהלך האימון. בפרויקט שלי הגדרתי את משתנה זה ל 0.001, זהו ערך קצב למידה נפוץ עבור משימות למידה עמוקות רבות והרבה ביצועים טובים בתרחישים שונים.

לסיכום, אופטימיזציה היא כלי אשר מקנה למודלים את היכולת ללמוד ולהתאים לנתונים בצורה מיטבית. היכולת למצוא את הפרמטרים המיטביים עוזרת למודל להפיק מידע רלוונטי מהנתונים ולתפקד בצורה טובה על נתונים חדשים שהוא לא ראה קודם.

פונקציית אקטיבציה

פונקציית האקטיבציה מחליטה אם יש להפעיל נוירון או לא על ידי חישוב הסכום המשוקלל והוספת הטיה נוספת אליו. מטרת פונקציית ההפעלה היא להכניס אי-ליניאריות לפלט של נוירון.

לרשת העצבית יש נוירונים הפועלים בהתאמה עם משקל, הטיה ותפקוד ההפעלה שלהם. ברשת עצבית, היינו מעדכנים את המשקלים וההטיות של הנוירונים על בסיס השגיאה במוצא. תהליך זה ידוע בשם חלחול לאחור. פונקציות ההפעלה מאפשרות את החלחול לאחור מכיוון שהשיפועים מסופקים יחד עם השגיאה לעדכון המשקולות וההטיות.

מטרת פונקציית האקטיבציה היא להכניס אי-ליניאריות לרשת עצבית מלאכותית ולהפיק פלט מאוסף ערכי קלט המוזנים לשכבה. רשת עצבית ללא פונקציית הפעלה היא בעצם רק מודל רגרסיה ליניארית. פונקציית האקטיבציה מבצעת את הטרנספורמציה הלא-לינארית לקלט מה שהופך אותו ליכול ללמוד ולבצע משימות מורכבות יותר.

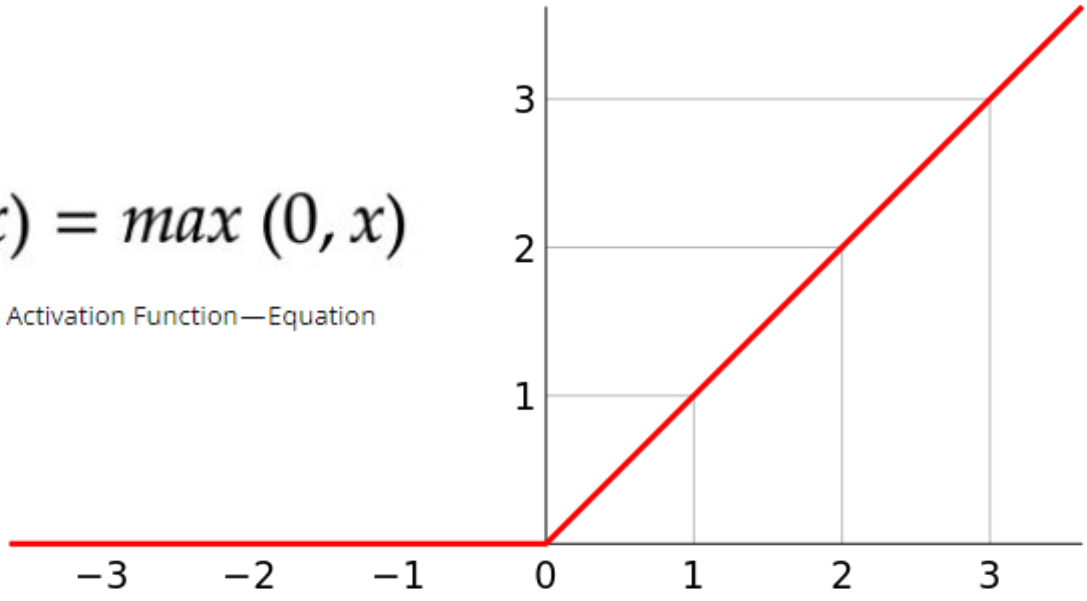
פונקציית האקטיבציה בפרויקט שלי היא – ReLU(Rectified Linear Unit).

ReLU היא פונקציית הפעלה בשימוש נרחב בלמידה עמוקה. זוהי פונקציה פשוטה אך רבת עוצמה המסייעת לשפר את הביצועים והיעילות של רשתות עצביות עמוקות.

הפונקציה נראית כך:

$$f(x) = max\ (0, x)$$

ReLU Activation Function—Equation



הפונקציה ReLU לוקחת קלט x ומחזירה את הערך החיובי שלו, או 0 אם הקלט שלילי. המשמעות היא שכל ערך לא שלילי עובר ללא שינוי (הנוירון מופעל), בעוד שערכים שליליים מוחלפים ב-0 (הנוירון אינו מופעל).

היתרון העיקרי של ReLU הוא ביכולתו להפעיל נוירונים בצורה ליניארית ויעילה. הפונקציה עוסקת ביעילות בבעיות הכרוכות בהספקים גבוהים ותכונות לא ליניאריות בנתונים, ומסייעת לזהות וללמוד פערים ומאפיינים חשובים בנתונים.

ReLU מקל על בעיית השיפוע הנעלם, שעלולה להתרחש בעת אימון רשתות עמוקות. הנגזרת של ReLU היא או 0 או 1, מה שהופך את הסיכוי שלה להרוויה או לגרום לשיפועים להיעלם במהלך החלחול לאחור. זה מאפשר אימון יציב ויעיל יותר של רשתות עצביות עמוקות.

לסיכום, פונקציית ReLU פשוטה ליישום ולחישוב, הדורשת השוואה ופעולה מקסימלית בלבד, היעילות החישובית שלה גבוהה בשל הליניאריות שלה והעובדה שהיא כוללת רק פעולות מתמטיות פשוטות, ReLU מציגה אי-לינאריות לרשת העצבית ומאפשרת לה ללמוד ולייצג קשרים מורכבים בנתונים, היא הראתה ביצועים טובים במשימות ראייה ממוחשבת, כגון סיווג תמונה וזיהוי אובייקטים ולכן בחרתי בפונקציה זו ליישום הפרויקט.

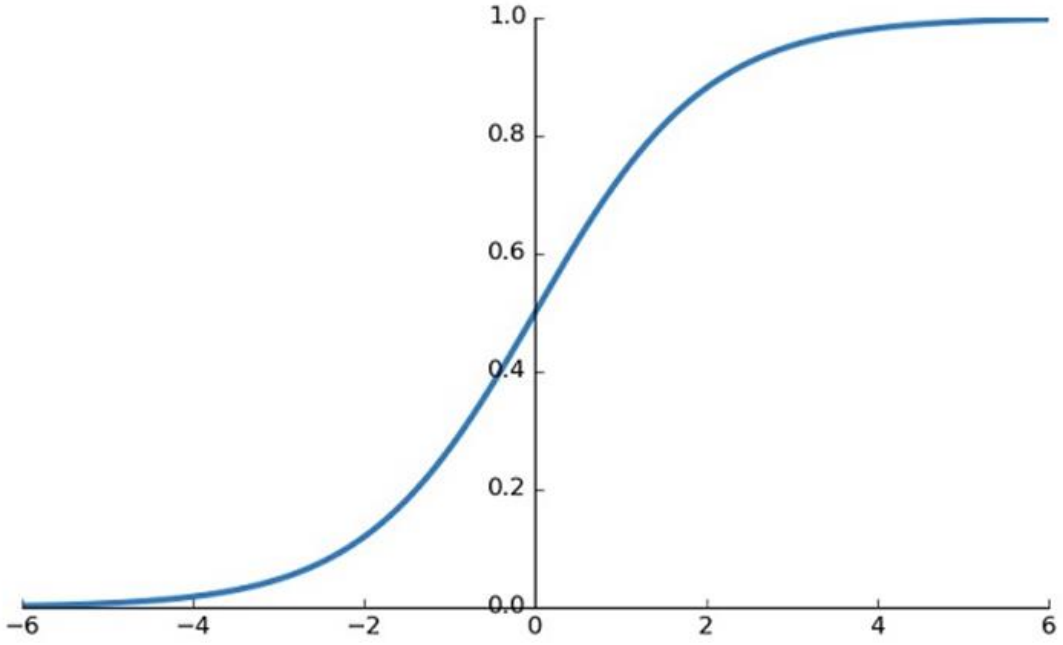
השכבה האחרונה ברשת משתמשת בפונקציית Softmax.

הפונקציה softmax היא פונקציה שהופכת וקטור של K ערכים אמיתיים לווקטור של K ערכים ממשיים המסכמים ל-1. ערכי הקלט יכולים להיות חיוביים, שליליים, אפס או גדולים מאחד, אבל ה-softmax הופך אותם לערכים בין 0 ו-1, כך שניתן לפרש אותם כהסתברויות. אם אחד הקלטים קטן או שלילי, ה-softmax הופך אותו להסתברות קטנה, ואם קלט גדול, אז הוא הופך אותו להסתברות גדולה, אבל הוא תמיד יישאר בין 0 ל-1.

פונקציית softmax נקראת לפעמים פונקציית softargmax, או רגרסיה לוגיסטית רב-מעמדית. הסיבה לכך היא שה-softmax הוא הכללה של רגרסיה לוגיסטית שניתן להשתמש בה לסיווג רב מחלקות, והנוסחה שלה דומה מאוד לפונקציה הסיגמואיד המשמשת לרגרסיה לוגיסטית. ניתן להשתמש בפונקציית softmax במסווג רק כאשר המחלקות סותרות זו את זו.

רשתות עצביות רבות-שכבתיות מסתיימות בשכבה לפני אחרונה אשר מפיקה ציונים בעלי ערך אמיתי שאינם מותאמים בצורה נוחה ואשר עשוי להיות קשה לעבוד איתם. כאן ה-softmax שימושי מאוד מכיוון שהוא ממיר את הציונים להתפלגות הסתברות מנורמלת, שניתן להציג למשתמש או להשתמש כקלט למערכות אחרות. מסיבה זו נהוג לצרף פונקציית softmax כשכבה הסופית של הרשת העצבית.

הפונקציה נראית כך:



$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

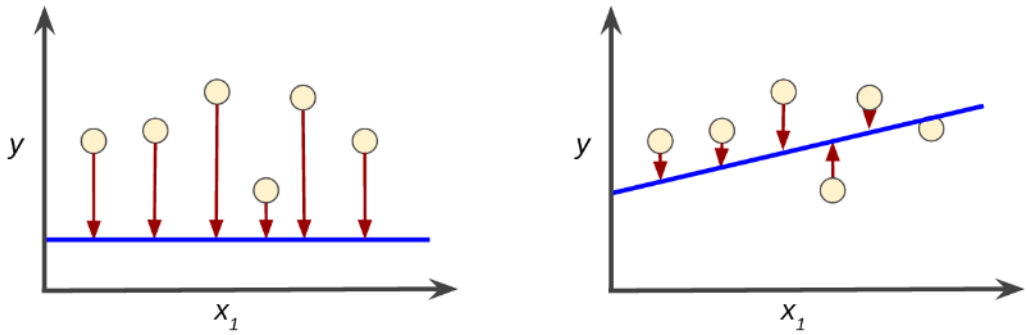
כאשר:

$\vec{z}$	וקטור הקלט לפונקציית softmax, מורכב מ-(z0, ... zK)
z_i	ערך של Z במקום i. כל ערכי ה-zi הם האלמנטים של וקטור הקלט לפונקציה softmax, והם יכולים לקחת כל ערך אמיתי, חיובי, אפס או שלילי. לדוגמה, רשת עצבית יכולה להוציא וקטור כגון (0.62-, 8.12, 2.53), שאינה התפלגות הסתברות חוקית, ומכאן הסיבה לכך שה-softmax יהיה הכרחי.
e^{z_i}	הפונקציה האקספוננציאלית הסטנדרטית מיושמת על כל אלמנט של וקטור הקלט. זה נותן ערך חיובי מעל 0, שיהיה קטן מאוד אם הקלט היה שלילי, וגדול מאוד אם הקלט היה גדול. עם זאת, הוא עדיין לא קבוע בטווח (0, 1) שזה מה שנדרש מהסתברות.
$\sum_{j=1}^{K} e^{z_j}$	מונח זה מבטיח שכל ערכי הפלט של הפונקציה יהיה בטווח (0, 1), ובכך יהווה התפלגות הסתברות חוקית.
K	מספר הסיווגים האפשריים.

פונקציית שגיאה

אימון מודל פירושו פשוט לימוד (קביעת) ערכים טובים עבור כל המשקולות וההטיה מדוגמאות מסומנות (מאגר הנתונים). בלמידה מפוקחת, אלגוריתם למידת מכונה בונה מודל על ידי בחינת דוגמאות רבות וניסיון למצוא מודל שממזער אובדן.
הפסד הוא העונש על תחזית גרועה. כלומר, הפסד הוא מספר המציין כמה גרועה הייתה התחזית של המודל בדוגמה בודדת. אם התחזית של המודל מושלמת, ההפסד הוא אפס; אחרת, ההפסד גדול יותר. המטרה של אימון מודל היא למצוא קבוצה של משקולות והטיות עם אובדן נמוך, בממוצע, על פני כל הדוגמאות.

הגרפים הבאים מציגים מודל הפסד גבוה משמאל ומודל הפסד נמוך מימין כאשר הקו הכחול הוא החיזוי והחיצים מסמלים את ההפסד.



החיצים בגרף השמאלי ארוכים בהרבה מהחיצים בגרף הימני. ברור שהקו בגרף הימני הוא מודל חיזוי הרבה יותר טוב מהקו בשמאלי.
פונקציית השגיאה חשובה מכיוון שאם הערך של פונקציית ההפסד נמוך יותר אז זה מודל טוב אחרת, עלינו לשנות את הפרמטר של המודל ולמזער את ההפסד.

בפרויקט שלי השתמשתי בפונקציית "categorical_crossentropy". פונקציה זו נפוצה בשימוש בבעיות סיווג מרובות מחלקות שבהן המחלקות סותרות זו את זו. היא נמצאת בשימוש נרחב במודלים של למידה עמוקה, במיוחד במשימות כמו סיווג תמונות וזיהוי אובייקטים.
פונקציית האובדן "categorical_crossentropy" מודדת את השונות בין התפלגות המחלקה האמיתית להסתברויות המחלקות החזויות. היא מחשבת את האובדן בין התוויות האמיתיות לבין ההסתברויות החזויות. המטרה היא למזער את ההפסד הזה במהלך תהליך האימון, מה שעוזר למודל ללמוד לבצע תחזיות מדויקות יותר.

כך פועלת הפונקציה:

1. התוויות האמיתיות מיוצגות בתור וקטורים. לכל וקטור תווית אמיתי יש אורך השווה למספר המחלקות, עם ערך של 1 עבור המחלקה האמיתית ו-0 עבור כל המחלקות האחרות. לדוגמה, אם יש שלוש מחלקות, תווית אמיתית של מחלקה 2 תוצג כ-[0, 1, 0].
2. המודל מוציא הסתברויות חזויות עבור כל מחלקה. הסתברויות אלו מתקבלות משכבת ההפעלה הסופית של softmax במודל. ההסתברויות החזויות עבור כל מחלקה מסתכמות ב-1.
3. ההפסד מחושב על ידי השוואת וקטורי התווית האמיתיים עם ההסתברויות החזויות. הוא מודד את השוני בין שתי ההפצות. ערך ההפסד גבוה יותר כאשר ההסתברויות החזויות חורגות יותר מההתפלגות התווית האמיתית.
4. במהלך האימון, המודל מתאים את המשקולות וההטיות שלו באמצעות טכניקות אופטימיזציה של ירידה בשיפוע כדי למזער את האובדן. אלגוריתם החלחול לאחור מחשב את ההדרגות ומעדכן את הפרמטרים של המודל כדי לשפר את התחזיות.

פונקציה זו נמצאת בשימוש נרחב עבור בעיות סיווג מרובה כיתות והיא יעילה באימון מודלים לביצוע תחזיות מדויקות על פני מספר מחלקות.

רגולציה במודל

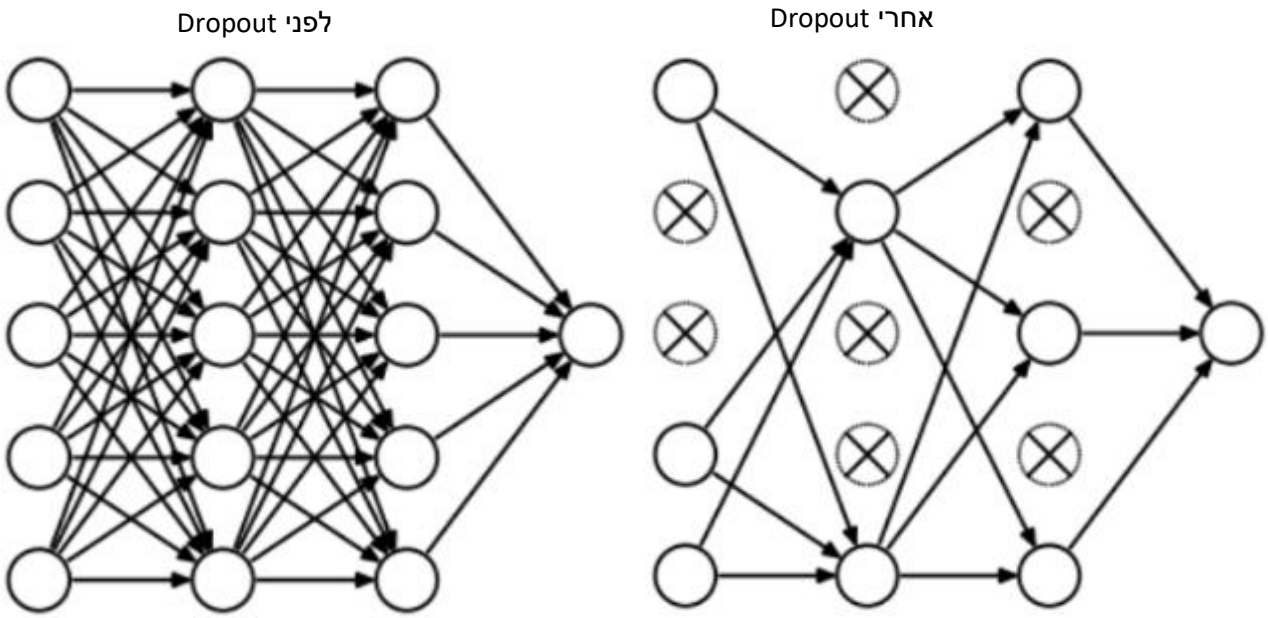
רגוליזציה היא טכניקה המשמשת למידת מכונה ולמידה עמוקה כדי למנוע התאמת יתר ולשפר את ביצועי ההכללה של מודל. כאשר מודל מתאים את עצמו קרוב מדי לנתוני האימון, הוא עלול לסבול מהתאמת יתר, כלומר המודל לומד לזהות פרטים ספציפיים בנתוני האימון והופך פחות גמיש בזיהוי דפוסים חדשים.

רגולציה מתבצעת על ידי הוספת תנאים לפונקציית ההפסד של המודל במהלך תהליך האימון. שני סוגים נפוצים של טכניקות רגוליזציה הם:

- L1 Regularization: בטכניקה זו, מתווסף תנאי לפונקציית ההפסד של המודל שהוא פרופורציונלי ישירות לערכים האבסולוטיים (ערך מוחלט) של הפרמטרים. כתוצאה מכך, תנאי זה מעודד צמיחת מודלים דלילים ומקדם בחירת משתנים חשובים במודל.
- L2 Regularization: בטכניקה זו מתווסף תנאי לפונקציית ההפסד של המודל שהוא פרופורציונלי לערכי הריבוע של הפרמטרים. שימוש בטכניקה זו נוטה להקטין את הערכים הגבוהים של הפרמטרים, מה שמוביל למודלים עם פחות דגש על משתנים פחות חשובים.

שתי שיטות אלה מגדילים את המרחק בין נתוני האימון למשקולות המופיעות במודל, עוזרים לדגם להתמודד עם התאמה יתר ומפחיתים את השגיאה הנגרמת מהסתמכות על מקרי האימון בלבד.

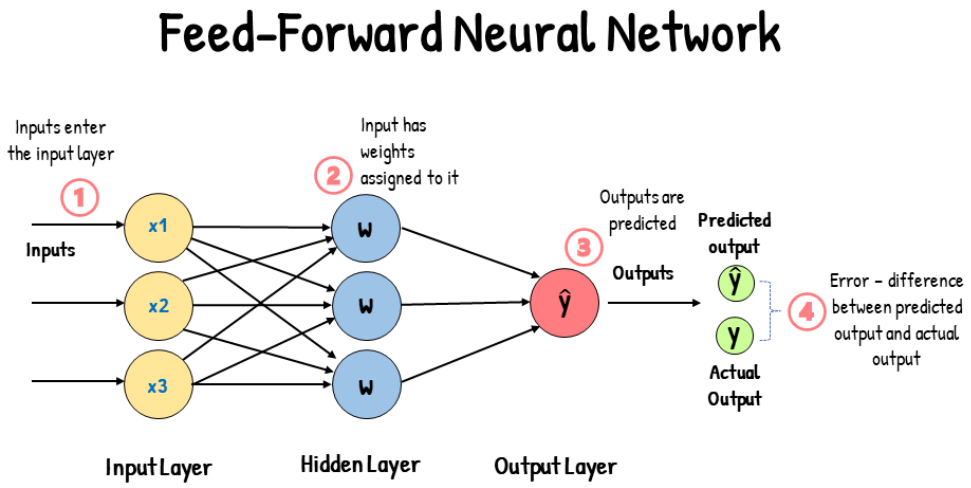
השיטה בה השתמשתי במודל שלי היא שכבת Dropout. Dropout כרוכה בנשירה אקראית של חלק הנוירונים במהלך האימון על ידי הגדרת התפוקות שלהם לאפס. אקראיות זו מאלצת את הרשת ללמוד ייצוגים חזקים ועצמאיים יותר. Dropout מפחיתה את התלות ההדדית של נוירונים, ומעודדת אותם ללמוד תכונות שימושיות בנפרד. Dropout היא טכניקה רבת עוצמה המסייעת במניעת התאמת יתר, מקדמת עצמאות תכונה ומשפרת את ההכללה של מודלים של למידה עמוקה. הוא מציג אקראיות ופועל כמכלול של רשתות משנה, מה שהופך את המודל לחזק יותר ופחות תלוי בתאי עצב או תכונות ספציפיות.



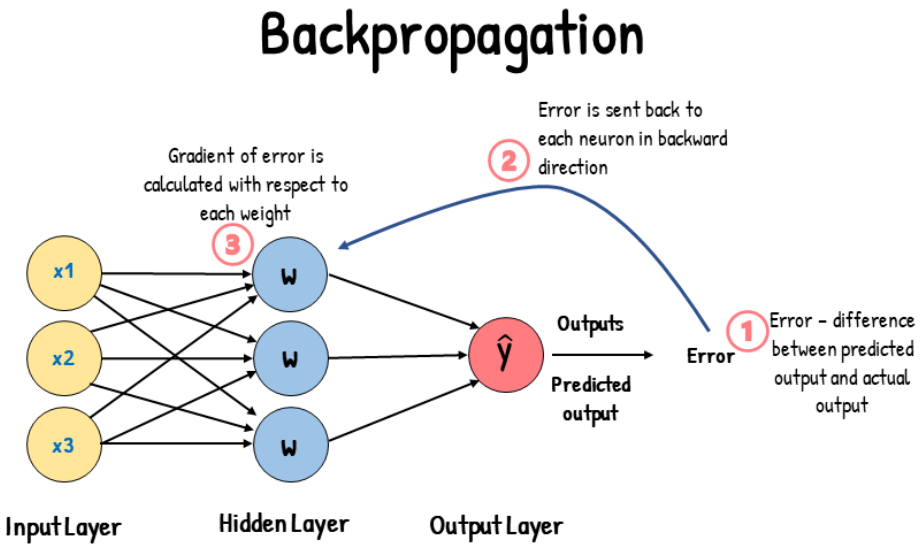
רגוליזציה היא כלי חשוב בלמידה עמוקה שמטרתה לשפר את ההכללה והחוסן של המודל לנתונים חדשים שלא היו חלק ממערך ההדרכה. בעזרתה מודלים יכולים להשיג ביצועים טובים יותר בזיהוי דפוסים כלליים ולטפל ביעילות במגוון רחב של מקרים.

חלחול אחורה וקדימה

- חלחול קדימה (Feed Forward) – מתייחס לתהליך של הפצת נתוני קלט דרך הרשת העצבית כדי לקבל פלט חיזוי. בפרויקט זה, נתוני הקלט מורכבים מתמונות של עלים בשלושה מצבים שונים, וחיזוי הפלט הוא סיווג המצב של הצמח. תהליך ההזנה כרוך בהעברת תמונות הקלט דרך שכבות הרשת העצבית, כאשר כל שכבה מבצעת סדרה של פעולות מתמטיות על נתוני הקלט כדי ליצור הפעלות. הפעלות אלו מועברות לאחר מכן לשכבות הבאות עד לשכבה הסופית, אשר מייצרת את פלט החיזוי.



- חלחול לאחור (Back Propagation) – אלגוריתם המשמש לאימון הרשת העצבית על ידי התאמת המשקולות וההטיות שלה בהתבסס על השגיאות בין הפלטים החזויים לבין התוויות האמיתיות. זה כולל שני שלבים עיקריים: חישוב השיפוע של פונקציית ההפסד ביחד לפרמטרים של הרשת (המשקולות וההטיות) ושימוש בשיפוע זה לעדכון הפרמטרים באופן שימזער את ההפסד.



ImageDataGenerator

`ImageDataGenerator` הוא קלאס המסופק על ידי ספריית Keras המאפשר לבצע הגדלת נתונים בזמן אמת ועיבוד מקדים על נתוני תמונה במהלך האימון.
בעזרת קלאס זה אפשר להגדיל, לסובב, להזיז ולהפוך את אותן תמונות מהמאגר שלנו.

שיטה זו מרחיבה את גודל מערך הנתונים ומאפשרת למודל להכליל יותר טוב נתונים בתמונות.

במהלך האימון, ה-ImageDataGenerator משומש בדרך כלל עם שיטת flow_from_directory(),
ובה גם השתמשתי בפרויקט שלי.

flow_from_directory() – פונקציה המסופקת על ידי המחלקה `ImageDataGenerator` ב-Keras.
היא משמשת ליצירת קבוצות של נתוני תמונה מוגדלים ממבנה ספריות. מבנה הספריות צריך לעקוב אחר פורמט מסוים שבו כל ספריית משנה מייצגת סוג או קטגוריה שונה של תמונות. שיטת flow_from_directory() מסמנת אוטומטית את התמונות על סמך שמות ספריית המשנה ומקצה את תוויות המחלקות המתאימות.

ImageDataGenerator ו- flow_from_directory() בקוד שלי –

```
train_datagen=ImageDataGenerator(rescale=1/255)
training_set = train_datagen.flow_from_directory(
    train_path,
    target_size= (150,150),
    batch_size = 48,
    class_mode = 'categorical')

test_datagen=ImageDataGenerator(rescale=1/255)
test_set = test_datagen.flow_from_directory(
    test_path,
    target_size=(150, 150),
    batch_size = 48,
    class_mode = 'categorical')
```

בקוד זה, `ImageDataGenerator` ו`flow_from_directory()` משמשים לעיבוד מוקדם של נתונים ויצירת קבוצות של נתונים מוגדלים להדרכה ובדיקת המודל.

ישנם שתי מופעים של `ImageDataGenerator`, בtrain_datagen המשמש לעיבוד מקדים של תמונות האימון, וגם בtest_datagen המשמש לעיבוד מקדים של תמונות הבדיקה.
בשני מופעים אלה, הפרמטר rescale=1/255 משנה את ערכי הפיקסלים של התמונות לטווח [0, 1].
נורמליזציה זו חשובה מכיוון שהיא עוזרת למודל להתכנס מהר יותר ומונעת מהשיפוע להיות גדול מדי.

שיטת `flow_from_directory` נקראת גם במופעי `train_datagen` וגם `test_datagen` כדי ליצור קבוצות של תמונות מוגדלות והתוויות המתאימות שלהן מהספריות שצוינו (healthy, rust, powdery).

פירוט הפרמטרים שנשלחו לשיטת "flow_from_directory":

- "train_path" ו- "test_path" (כל אחד בפעולה אחרת) – זהו הנתיב לספרייה המכילה את תמונות האימון.
- "target_size=(150,150)" – מציין את הגודל הרצוי שאליו ישתנו גודל התמונות.
- "batch_size" – זה מציין את מספר התמונות בכל אצווה שתיווצר במהלך האימון. בפרויקט שלי גודל פרמטר זה (בשתי הפעולות) הוא 48, כלומר שיטה זו תיצור אצווה של 48 תמונות כל פעם.
- "class_mode" – פרמטר זה מציין את סוג התוויות שיוחזרו על ידי המחולל. במקרה זה הוא מוגדר להיות 'categorical', משמע המחולל יחזיר עבור כל תמונה תווית מוחלטת. זה מתאים כאשר יש לך מספר קטגוריות ואתה רוצה להכשיר מודל לסיווג רב קטגורי.

שיטת `flow_from_directory` מסיקה אוטומטית את מספר המחלקות מספריות המשנה בתוך הספריות. זה יוצר קבוצות של תמונות מוגדלות יחד עם התוויות המתאימות להן לאימון המודל.

על ידי שימוש ב-'ImageDataGenerator' ו-'flow_from_directory', אפשר לעבד מראש ביעילות את נתוני התמונות וליצור קבוצות נתונים מוגדלות שיכולות לשמש להדרכה והערכת המודל.

שלב היישום:

תיאור והסבר כיצד היישום משתמש במודל

על מנת לחזות ולסווג תמונות בעזרת במודל בניתי פעולה **output()** שמטרתה לתת סיווג לתמונה שהיא מקבלת.

האלגוריתם –

```
def output(image_path, class_indices):
    img = load_img(image_path, target_size=(150, 150))
    img_array = img_to_array(img)
    img_array = img_array/255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Use the model to predict the class probabilities of the input image
    class_probabilities = model.predict(img_array)[0]

    # Convert the class probabilities to a class label
    predicted_class_index = np.argmax(class_probabilities)
    predicted_class = class_indices[predicted_class_index]

    print('Predicted class:', predicted_class)
    return predicted_class
```

מטרת הפעולה היא לחזות את תווית הקטגוריה עבור תמונת קלט באמצעות המודל המאומן. הפעולה מקבלת שתי משתנים :

1. `image_path`: נתיב לקובץ של תמונת הקלט שיש לסווג.
2. `class_indices': מילון הממפה בין אינדקסי הקטגוריות לתוויות הקטגוריות.

הפעולה פועלת כך:

1. היא טוענת את התמונה באמצעות הפונקציה `load_img` מהמודול `tensorflow.keras.preprocessing.image` ומשנה את גודלה ל 150*150 פיקסלים.
2. התמונה מומרת למערך NumPy באמצעות הפונקציה `img_to_array' מתוך `tensorflow.keras.preprocessing.image` וערכי הפיקסלים עוברים קנה מידה לטווח [0, 1] על ידי חלוקה ב-255.0.
3. המערך של התמונה מתרחב בציר האינדקס הראשון באמצעות `np.expand_dims`. זה הכרחי כדי להתאים לצורת הקלט שהמודל מצפה לו.
4. פונקציית `predict` של המודל משמשת להשגת הסתברויות המחלקה עבור תמונת הקלט. ההסתברויות החזויות מאוחסנות במשתנה `class_probabilities`.
5. הקטגוריה המנובאת נקבעת על ידי חיפוש האינדקס של הערך המרבי במערך `class_probabilities` באמצעות `np.argmax`.
6. התווית של הקטגוריה המנובאת מתקבלת מהמילון `class_indices' תוך שימוש באינדקס של הקטגוריה המנובאת.
7. התווית של הקטגוריה המנובאת מודפסת למסך.
8. לבסוף התווית של הקטגוריה המנובאת מוחזרת כפלט מהפונקציה.

תיאור הטכנולוגיה שעל פיה מומש ממשק המשתמש

על מנת לממש ממשק משתמש יצרתי חלון גרפי בעזרת ספריית tkinter.

- **ייבוא הספריות המתאימות ליצירת החלון הגרפי:**

```
import tkinter as tk
from tkinter import filedialog
from PIL import ImageTk, Image
```

- **בניית חלון הפתיחה ('start_window'):**

קטע קוד זה יוצר את החלון הפתיחה, מגדיר אותו לגודל 300*600 ונותן לו שם, לחלון זה קראתי – "Plant Disease Recognition – Kfir Eitan – Start Window".

```
start_window = tk.Tk()
start_window.geometry("600x300")
start_window.title("Plant Disease Recognition - Kfir Eitan - Start
Window")
```

- **תכולת חלון הפתיחה:**

בחלון הפתיחה יש כותרת המברכת את המשתמש וכפתור "Start" אשר כאשר ילחץ תופעל פעולת "open_prediction_window" אשר פותחת את החלון המרכזי בו יהיה את האפשרות לערוך סיווג לפי הקטגוריות.

```
lblNum = tk.Label(start_window, text="Welcome to plant disease
recognition", height=2, font=("Arial", 24))
lblNum.pack(side="top")

start_button = tk.Button(start_window, text="Start",
command=open_prediction_window, width=10, font=("Arial", 15))
start_button.pack(pady=50)

start_window.mainloop()
```

- **חלון החיזוי:**

פעולה זו יוצרת את חלון "prediction_window" בגודל 750*800, ונותנת לו את השם "Plant Disease Recognition – Kfir Eitan – Prediction Window". היא יוצרת בחלון שלושה כפתורים – כפתור המאפשר לעלות תמונה חדשה (upload), הוא פותח תיבת דו שיח ששם אפשר להעלות קבצים מהמחשב האישי של המשתמש, כפתור המוחק את התמונה שעלתה ואת החיזוי שהודפס (reset), כפתור שסוגר את החלון (exit) וגם מוסיפה כותרת המזמינה את המשתמש להעלות תמונה.
בנוסף, הפעולה יוצרת קנבס בגודל 500*500 בחלון, כאשר המשתמש יעלה תמונה היא תופיעה בקנבס הזה.

```
def open_prediction_window():
    global prediction_window
    global canvas
    global prediction_label
    global img

    start_window.destroy()   # Close the start window
    prediction_window = tk.Tk()
    prediction_window.geometry("750x800")# determines the size of the window
    prediction_window.title("Plant Disease Recognition - Kfir Eitan -
                            Prediction Window")# add title to the window
    lblNum = tk.Label(prediction_window, text="Please upload
                      picture:", height=2, font=("Arial", 24))
    lblNum.pack(side="top")

    browse_button = tk.Button(prediction_window, text="Upload",
                              command=browse_image, width=10,
                              font=("Arial", 15))
    browse_button.pack(pady=10)  # Increase the vertical spacing

    reset_button = tk.Button(prediction_window, text="Reset",
                             command=reset_image, width=10, fg='red',
                             font=("Arial", 15))
    reset_button.pack(pady=10)# Increase the vertical spacing

    canvas = tk.Canvas(prediction_window, width=500, height=500)
    canvas.pack()

    prediction_label = tk.Label(prediction_window, text="Prediction:"
                                , font=("Arial", 15))
    prediction_label.pack()

    exit_button = tk.Button(prediction_window, text="Exit",
                            command=exit_prediction_window, width=10,
                            font=("Arial", 15))
    exit_button.place(x=20, y=750)#Position the button at the bottom left
corner

    prediction_window.mainloop()
```

קוד הקולט את הDATA שעליו יבוצע החיזוי והתאמתו למבנה נתונים המתאים לחיזוי

בכדי שהמשתמש יעלה תמונה שתישלח לסיווג, עליו ללחוץ על כפתור "upload".
כאשר הוא ילחץ על כפתור זה הקטע קוד הבא יתחיל לפעול –

```
def browse_image():
    global img
    global canvas
    global prediction_label

    filename = filedialog.askopenfilename(initialdir="/",
               title="Select Image",
               filetypes=(("Image Files", "*.jpg *.jpeg *.png"),))

    if filename:
        img = Image.open(filename)
        img = img.resize((500,500))# Adjust the size of the displayed image
        img = ImageTk.PhotoImage(img)
        canvas.image = img # Save reference to keep the image object alive
        canvas.create_image(0, 0, anchor="nw", image=img)
        prediction_label.config(text="Prediction: " +
                                output(filename, class_indices))
```

פעולה זה מופעלת כאשר המשתמש לוחץ על כפתור "upload" בחלון "prediction_window". כאשר פעולה זו מתחילה לעבוד היא פותחת חלון דו שיח המאפשר למשתמש להעלות תמונה מהמחשב שלו. ברגע שהמשתמש בוחר תמונה הפעולה מאחזרת את נתיב הקובץ של התמונה הנבחרת.
לאחר מכן, הפעולה טוענת את התמונה הנבחרת באמצעות הפונקציה Image.open() מספריית PIL ומשנה את גודלה ל500*500 פיקסלים (הגודל של התמונה אשר תופיעה על המסך).
התמונה עולה על החלון, נשלחת לחיזוי על ידי "output()" ומדפיסה את הקטגוריה המנובאת על פני החלון הגרפי.

כאשר התמונה תגיע לפעולת output() גודלה ישתנה והיא תהיה בגודל 150*150 (לצורך הסיווג, המודל מצפה לנתון זה), בנוסף היא תומר למערך חד ממדי וערכי הפיקסלים עוברים קנה מידה לטווח [0, 1] על ידי חלוקה ב-255.0.

רק לאחר שהתמונה תעבור את כל השלבים הללו, תופעל עליה הפעולה
`model.predict(img_array)[0]` אשר משמשת להשגת הסתברויות המחלקה עבור תמונת הקלט.

מדריך למפתח:

תוכן הקוד

```
from PIL import Image
import random
import matplotlib.pyplot as plt
import os as os
import numpy as np
import tensorflow as tf
from tensorflow.keras.layers import LeakyReLU, Dense, Dropout,
Flatten, Input, BatchNormalization, Activation, Conv2D, MaxPooling2D,
GlobalAveragePooling2D
from tensorflow.keras.models import Model, Sequential, load_model
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# ---------------------------------------------------------------------
# CONNECTING DATA
print("---CONNECTING DATA---") # connect the DATA

# Set the directories for training and testing images
train_healthy_dir = 'data/Train/Healthy'
train_powdery_dir = 'data/Train/Powdery'
train_rust_dir = 'data/Train/Rust'

test_healthy_dir = 'data/Test/Healthy'
test_powdery_dir = 'data/Test/Powdery'
test_rust_dir = 'data/Test/Rust'

# Set the path for validation, training, test, and real directories
val_path = 'data/Validation'
train_path = 'data/Train'
test_path = 'data/Test'
real_path = 'data'

print("---finished to connect---")
print("")

# ---------------------------------------------------------------------
# FILE PATHS
# Print the file paths to verify the loaded directories
print("files path: ")
for dirname, _, filenames in os.walk(real_path):
    print(dirname)
print("")

# ---------------------------------------------------------------------
# SHUFFLE FILE NAMES
# Shuffle the file names in the training and testing directories for
each category

train_healthy_names = os.listdir(train_healthy_dir)
random.shuffle(train_healthy_names)
print(train_healthy_names[:10])

train_powdery_names = os.listdir(train_powdery_dir)
random.shuffle(train_powdery_names)
print(train_powdery_names[:10])
```

```
train_rust_names = os.listdir(train_rust_dir)
random.shuffle(train_rust_names)
print(train_rust_names[:10])


test_healthy_hames = os.listdir(test_healthy_dir)
print(test_healthy_hames[:10])

test_powdery_names = os.listdir(test_powdery_dir)
print(test_powdery_names[:10])

test_rust_names = os.listdir(test_rust_dir)
print(test_rust_names[:10])


real_names = os.listdir(real_path)
print(real_names)

print("")

# ------------------------------------------------------------------
# RESIZE IMAGES
# Resize the images to a size of 150x150 pixels

print("start to change the size of the picture", end='s')

small = (150, 150)
category_names = os.listdir(train_path)

small = (150, 150)
category_names = os.listdir(train_path)

# Resize training images
for category in category_names:
    category_path = os.path.join(train_path, category)
    for file in os.listdir(category_path):
        if file.endswith('.jpg') or file.endswith('.png'):
            img_path = os.path.join(category_path, file)
            with Image.open(img_path) as img:
                img.thumbnail(small)
                img.save(img_path)

# Resize testing images
category_names = os.listdir(test_path)
for category in category_names:
    category_path = os.path.join(test_path, category)
    for file in os.listdir(category_path):
        if file.endswith('.jpg') or file.endswith('.png'):
            img_path = os.path.join(category_path, file)
            with Image.open(img_path) as img:
                img.thumbnail(small)
                img.save(img_path)


print("-----> Done;")
print("")

# ------------------------------------------------------------------
```

```
# DATA GENERATION
# Create the ImageDataGenerator for training and testing data
train_datagen=ImageDataGenerator(rescale=1/255)
training_set = train_datagen.flow_from_directory(
    train_path,
    target_size= (150,150),
    batch_size = 48,
    class_mode = 'categorical')

test_datagen=ImageDataGenerator(rescale=1/255)
test_set = test_datagen.flow_from_directory(
    test_path,
    target_size=(150, 150),
    batch_size = 48,
    class_mode = 'categorical')

# ---------------------------------------------------------------------
# MODEL ARCHITECTURE
# Create the model architecture

print("---START TO BUILD THE MODEL---")
print("")

# Create the model
model = Sequential()

# Add convolutional layers
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu',
input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten()) # Flatten the output tensor from the
convolutional layers

# Add dense layers for classification
model.add(Dense(units=512, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(units=3, activation='softmax'))


# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Print a summary of the model architecture
model.summary()

print("")

# ---------------------------------------------------------------------
```

```
# MODEL TRAINING
# Train the model
print("---START TRAIN THE MODEL---")
history = model.fit(training_set,
                    validation_data=test_set,
                    epochs=10,
                    steps_per_epoch=len(training_set),
                    validation_steps=len(test_set))
print("")

# ------------------------------------------------------------------
# MODEL EVALUATION
# Evaluate the model on the test set
model.evaluate(test_set)

#model.save("PlantDiseaseModel.h5")
# ------------------------------------------------------------------
# PLOT ACCURACY
#accurracy VS validation accuracy

# Get the accuracy and validation accuracy values from the history
object
accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

# Set the figure size and font style
plt.figure(figsize=(8, 6))
plt.rcParams.update({'font.size': 14})

# Plot the accuracy and validation accuracy curves
plt.plot(accuracy, linewidth=2, color='red')
plt.plot(val_accuracy, linewidth=2, color='green')

# Set the plot title, labels, and legend
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'], loc='lower right')

# Add grid lines and tighten the layout
plt.grid(True)
plt.tight_layout()

# Show the plot
print("open graph - Model Accurac", end='y')
plt.show()
print("-------> close;")
print("")

# ------------------------------------------------------------------
# PLOT LOSS
#loss VS validation loss

# Get the loss and validation loss values from the history object
loss = history.history['loss']
val_loss = history.history['val_loss']

# Set the figure size and font style
plt.figure(figsize=(8, 6))
plt.rcParams.update({'font.size': 14})
```

```
# Plot the loss and validation loss curves
plt.plot(loss, linewidth=2, color='red')
plt.plot(val_loss, linewidth=2, color='green')

# Set the plot title, labels, and legend
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper right')

# Add grid lines and tighten the layout
plt.grid(True)
plt.tight_layout()

# Show the plot
print("open graph - Model Los", end='s')
plt.show()
print("-------> close;")
print("")

# ---------------------------------------------------------------------
# PREDICTION FUNCTION
def output(image_path, class_indices):
    img = load_img(image_path, target_size=(150, 150))
    img_array = img_to_array(img)
    img_array = img_array/255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Use the model to predict the class probabilities of the input
image
    class_probabilities = model.predict(img_array)[0]

    # Convert the class probabilities to a class label
    predicted_class_index = np.argmax(class_probabilities)
    predicted_class = class_indices[predicted_class_index]

    print('Predicted class:', predicted_class)
    return predicted_class

# Define the class indices for prediction
class_indices = {0: 'healthy', 1: 'powdery', 2: 'rust'}
```

```
# USER INTERFACE
# Create a user interface to browse and predict images

# Import the necessary libraries
import tkinter as tk
from tkinter import filedialog
from PIL import ImageTk, Image


# Function to browse and display an image
def browse_image():
    global img
    global canvas
    global prediction_label

    # Open a file dialog to select an image file
    filename = filedialog.askopenfilename(initialdir="/",
title="Select Image",
                                          filetypes=(("Image Files",
"*.jpg *.jpeg *.png"),))
    if filename:
        # Open and resize the selected image
        img = Image.open(filename)
        img = img.resize((500, 500))
        img = ImageTk.PhotoImage(img)

        # Update the canvas with the new image
        canvas.image = img
        canvas.create_image(0, 0, anchor="nw", image=img)

        # Make a prediction using the selected image
        prediction_label.config(text="Prediction: " +
output(filename, class_indices))

# Function to reset the image and prediction label
def reset_image():
    canvas.delete("all")
    prediction_label.config(text="Prediction: ")

# Function to exit the prediction window
def exit_prediction_window():
    prediction_window.destroy()


# Function to exit the start window
def exit_start_window():
    start_window.destroy()

# Function to open the prediction window and start the application
def open_prediction_window():
    global prediction_window
    global canvas
    global prediction_label
    global img

    # Close the start window
    start_window.destroy()

    # Create the prediction window
    prediction_window = tk.Tk()
    prediction_window.geometry("750x800")
```

```
    prediction_window.title("Plant Disease Recognition - Prediction
Window")

    # Label to prompt the user to upload an image
    lblNum = tk.Label(prediction_window, text="Please upload a
picture:", height=2, font=("Arial", 24))
    lblNum.pack(side="top")

    # Button to browse and upload an image
    browse_button = tk.Button(prediction_window, text="Upload",
command=browse_image, width=10, font=("Arial", 15))
    browse_button.pack(pady=10)

    # Button to reset the image and prediction
    reset_button = tk.Button(prediction_window, text="Reset",
command=reset_image, width=10, fg='red',
                             font=("Arial", 15))
    reset_button.pack(pady=10)

    # Canvas to display the uploaded image
    canvas = tk.Canvas(prediction_window, width=500, height=500)
    canvas.pack()

    # Label to display the prediction result
    prediction_label = tk.Label(prediction_window, text="Prediction:
", font=("Arial", 15))
    prediction_label.pack()

    # Button to exit the prediction window
    exit_button = tk.Button(prediction_window, text="Exit",
command=exit_prediction_window, width=10,
                            font=("Arial", 15))
    exit_button.place(x=20, y=750)

    # Run the prediction window
    prediction_window.mainloop()

# Create the start window
start_window = tk.Tk()
start_window.geometry("600x300")
start_window.title("Plant Disease Recognition - Start Window")

# Label to welcome the user
lblNum = tk.Label(start_window, text="Welcome to Plant Disease
Recognition", height=2, font=("Arial", 24))
lblNum.pack(side="top")

# Button to start the application
start_button = tk.Button(start_window, text="Start",
command=open_prediction_window, width=10, font=("Arial", 15))
start_button.pack(pady=50)

# Button to exit the application
exit_button = tk.Button(start_window, text="Exit",
command=exit_start_window, width=10, font=("Arial", 15))
exit_button.pack(pady=10)

# Run the start window
start_window.mainloop()
```

פרמטרים חשובים בקוד לאימון המודל

תיקיות הDATA –

פרמטרים אלה מציינים את נתיבי התיקייה עבור קבוצות שונות של תמונות (training, testing, validation, real images). הם חשובים מכיוון שהם מגדירים את מיקום הנתונים שישמשו לאימון, אימות והערכה.
- `train_healthy_dir`: נתיב ספרייה לתמונות אימון של קטגוריית צמח בריא.
- `train_powdery_dir`: נתיב ספרייה לתמונות אימון של קטגוריית צמח אבקני.
- `train_rust_dir`: נתיב ספרייה לתמונות אימון של קטגוריית החלודה.
- `test_healthy_dir`: נתיב ספרייה לבדיקת תמונות של קטגוריית צמח בריא.
- `test_powdery_dir`: נתיב ספרייה לבדיקת תמונות של קטגוריית צמח אבקני.
- `test_rust_dir`: נתיב ספרייה לבדיקת תמונות של קטגוריית החלודה.
- `val_path`: נתיב ספרייה לתמונות אימות.
- `train_path`: נתיב ספרייה לתמונות אימון.
- `test_path`: נתיב ספריה לבדיקת תמונות.
- `real_path`: נתיב לספרייה המכילה את כל התמונות.

ארכיטקטורת המודל –

בשכבות הקונבולוציוניות:
(1) 'filters': קובע את מספר המסננים שיש בכל שכבה קונבולוציונית.
(2) 'karnel_size': קובע את הגודל של המסנן הקונבולוציוני.
(3) 'activation': קובע את פונקציית ההפעלה הנמצאת בשכבה קונבולוציונית.

פרמטרים אלה חשובים מכיוון שהם קובעים את המורכבות וההתנהגות של השכבות הקונבולוציוניות, החיוניות ללימוד תכונות מתמונות קלט.

בשכבות הצפופות (Dense):
(1) 'units': קובע את המורכבות של השכבה ומשפיע על יכולת הלמידה של המודל.
(2) 'activation': קובע את פונקציית ההפעלה הנמצאת בשכבה קונבולוציונית.

פרמטרים אלו מגדירים את מספר נוירונים ואת פונקציית ההפעלה המשמשת בשכבות הצפופות של הדגם. הם חשובים מכיוון שהם שולטים במורכבות ובהתנהגות של השכבות הצפופות, האחראיות לביצוע סיווגים על סמך התכונות הנלמדות.

בשכבת Dropout:
(1)'rate': פרמטר זה מציין את שיעור הנשירה של הנוירונים.

פרמטר זה חשוב כי נשירה (של נוירונים) מסייעת במניעת התאמת יתר ומשפרת את יכולת ההכללה של המודל.

יצירת נתונים (Data Generation) –

(1) 'rescale': משנה את ערכי הפיקסלים של התמונות לטווח [0, 1].

פרמטר זה חשוב כי שינוי קנה מידה עוזר לנרמל את נתוני הקלט לטווח משותף, מה שמקל על המודל ללמוד ולהתכנס במהלך האימון.

אימון המודל –

(1) 'epochs': פרמטר זה קובע את מספר הפעמים שהמודל יחזור על כל מערך האימון. זה חשוב מכיוון שהוא שולט על משך האימון הכולל ועל מספר העדכונים שהמודל יקבל.
(2) 'steps_per_epoch': מספר השלבים שיש לעבד במהלך כל אפוק. פרמטר זה חשוב לחלוקת מערך האימון לקבוצות קטנות יותר, מה שעוזר בעדכון יעיל של משקלי המודל ומעקב אחר התקדמות האימון.
(3) 'validation_steps': פרמטר זה קובע את מספר שלבי האימות שיש לעבד לאחר כל אפוק במהלך אימון המודל. זה חשוב להערכת ביצועי המודל על מערך האימות ולניטור יכולת ההכללה שלו.

פונקציית החיזוי ()output –

(1) 'image_path': פרמטר זה מציין את הנתיב של התמונה שעבורה המודל יבצע סיווג. פרמטר זה חשוב כי הוא מגדיר את תמונת הקלט שעליה יתבסס חיזוי המודל.
(2) 'class_indices': פרמטר זה הוא מילון הממפה מדדי מחלקות לתוויות מחלקות. זה חשוב מכיוון שהוא עוזר לפרש את חיזוי המודל על ידי מתן המיפוי בין מדדי מחלקות מספריים והתוויות המתאימות להם.

פרמטרים חשובים בקוד ליצירת החלונות הגרפיים

גודל החלון –

(1) 'geometry': מציין את הגודל והמיקום של החלון. משתנה זה קובע כיצד יוצג החלון על המסך, כולל הרוחב, הגובה והמיקום שלו.

כותרת החלון –

(1) 'title': מגדיר את כותרת החלון. זה חשוב מכיוון שהוא מספק שם או תיאור משמעותיים לחלון.

תוויות –

(1) 'text': מגדיר את תוכן הטקסט של ווידג'ט של תווית. תוויות משמשות להצגת טקסט סטטי או הודעות למשתמש. פרמטר זה חשוב מכיוון שהוא מגדיר את התוכן שיוצג בתווית, מספק מידע או הוראות למשתמש.

כפתורים –

(1) 'text': מגדיר את הטקסט המוצג על הכפתור. זה חשוב מכיוון שהוא מציין מטרתו או פעולתו של הכפתור.
(2) 'command': מציינת את הפונקציה או הפעולה שיש לבצע בעת לחיצה על הכפתור. פרמטר זה חשוב מכיוון שהוא מגדיר את הפונקציונליות הקשורה לכפתור. בעת לחיצה על הכפתור, הפונקציה או הפעולה שצוינו תופעל.
(3) 'widht': מציין את רוחב הווידג'ט של הכפתור. פרמטר `רוחב` חשוב לשליטה בגודל ובמראה הכפתור. זה עוזר בעיצוב ממשק נעים ויזואלי שמיש.
(4) 'font': מציין את סגנון הגופן והגודל עבור טקסט הלחצן. הפרמטר הזה חשוב להתאמה אישית של הסגנון החזותי של טקסט הכפתור.

קנבס –

(1) 'widht': מציין את הרוחב של הקנבס. זה חשוב מכיוון שהוא קובע את רוחב אזור הקנבס שבו ניתן להציג את התמונה שהמשתמש העלה. זה מספק שליטה על הגודל והפריסה של הקנבס בתוך החלון.
(2) 'height': מציין את גובה הקנבס. פרמטר `גובה` חשוב לקביעת גובה שטח הקנבס. זה מאפשר להגדיר את המרחב האנכי הזמין להצגת התמונה בתוך הקנבס.

תיבת דו שיח –

(1) 'initialdir': מציין את הספרייה הראשונית המוצגת בתיבת הדו-שיח של הקובץ. הפרמטר הזה חשוב מכיוון שהוא מגדיר את ספריית ההתחלה כאשר תיבת הדו-שיח של הקובץ נפתחת. הוא מספק דרך נוחה למשתמשים לנווט לספרייה ספציפית או לגשת למיקום המועדף עליהם לבחירת קבצים.
(2) 'title': מגדיר את הכותרת של חלון הדו-שיח של הקובץ. הפרמטר 'כותרת' חשוב מכיוון שהוא מספק כותרת תיאורית לתיבת הדו-שיח של הקובץ. זה עוזר למשתמשים להבין את מטרת הדו-שיח ואת סוג הקבצים שהם צפויים לבחור.
(3) 'filetypes': מציין את סוגי הקבצים המותרים שניתן לבחור בתיבת הדו-שיח. הפרמטר `filetypes` חשוב להגבלת סוגי הקבצים שניתן לבחור. זה עוזר לסנן פורמטים לא רצויים של קבצים ומבטיח שהמשתמש יכול לבחור רק קבצים התואמים לאפליקציה.

פונקציות משמעותיות בקוד

קוד המשנה את גודלי התמונות –

```
small = (150, 150)
category_names = os.listdir(train_path)

# Resize training images
for category in category_names:
    category_path = os.path.join(train_path, category)
    for file in os.listdir(category_path):
        if file.endswith('.jpg') or file.endswith('.png'):
            img_path = os.path.join(category_path, file)
            with Image.open(img_path) as img:
                img.thumbnail(small)
                img.save(img_path)

# Resize testing images
category_names = os.listdir(test_path)
for category in category_names:
    category_path = os.path.join(test_path, category)
    for file in os.listdir(category_path):
        if file.endswith('.jpg') or file.endswith('.png'):
            img_path = os.path.join(category_path, file)
            with Image.open(img_path) as img:
                img.thumbnail(small)
                img.save(img_path)
```

קוד זה מבצע עיבוד מקדים של תמונה על ידי שינוי גודל ושמירת התמונות בספריות המקוריות.

הקוד עובר בלולאה על התמונות בספריות הtest ו-train. לכל קובץ תמונה בכל קטגוריה (healthy, rust, powdery) בתוך הספריות הוא בודק אם לקובץ יש סיומת ".jpg" או ".png". אם כן, הוא פותח את התמונה באמצעות הפונקציה `Image.open()` של ספריית PIL. לאחר מכן, הוא משנה את גודל התמונה לגודל קטן יותר המוגדר על ידי המשתנה `small` (במקרה זה, (150, 150)) באמצעות שיטת `thumbnail()` של האובייקט PIL Image. שיטה זו שומרת על יחס הגובה-רוחב תוך שינוי גודל התמונה כך שיתאים למידות שצוינו. לבסוף, הוא שומר את התמונה שגודלה שונה בחזרה לנתיב התמונה המקורית, ומחליף את קובץ התמונה המקורי.

ישנן שתי פונקציות שעושות את אותה הפעולה אך על תיקיות שונות. הראשונה עוברת על תיקיית train והשני על test.

שינוי גודל התמונות לגודל קטן יותר משרת מספר מטרות:

- זה עוזר להפחית את הדרישות החישוביות במהלך אימון מודלים. תמונות קטנות יותר דורשות פחות זיכרון וכוח עיבוד, מה שמאפשר אימון מהיר יותר.
- כאשר מאמנים מודל למידה עמוקה, מקובל לקבל תמונות קלט באותו גודל. שינוי גודל התמונות מבטיח שיש להן ממדים עקביים, דבר הכרחי להזנתן לדגם.
- זה יכול לשפר את ביצועי האימון. תמונות קטנות יותר יכולות לעזור להפחית את הסיכון להתאמת יתר מכיוון שהמודל צריך ללמוד מפחות פרטים ויכול להתמקד בתכונות החשובות ביותר.

קטע קוד זה חיוני להכנת הנתונים לפני הזנתו למודל. זה משנה את גודל התמונות לגודל קטן ועקבי יותר, ומבטיח אימון יעיל תוך שמירה על תכונות חשובות.

קוד המחלק את הdata –

```
# DATA GENERATION
# Create the ImageDataGenerator for training and testing data
train_datagen=ImageDataGenerator(rescale=1/255)
training_set = train_datagen.flow_from_directory(
    train_path,
    target_size= (150,150),
    batch_size = 48,
    class_mode = 'categorical')

test_datagen=ImageDataGenerator(rescale=1/255)
test_set = test_datagen.flow_from_directory(
    test_path,
    target_size=(150, 150),
    batch_size = 48,
    class_mode = 'categorical')
```

הפעולה ImageDataGenerator מספריית keras משמשת להפקת קבוצות של נתוני תמונה עם הגדלת נתונים בזמן אמת.
הפרמטר rescale=1/255 הוא ארגומנט המשמש לשינוי קנה מידה של ערכי הפיקסלים של התמונות. במקרה זה, ערכי הפיקסלים מחולקים ב-255, שהוא ערך הפיקסלים המרבי בתמונה. חלוקה ב-255 משנה את ערכי הפיקסלים לטווח שבין 0 ל-1, ולמעשה מנרמל את ערכי הפיקסלים.

שיטת "flow_from_directory" היא מופע של `ImageDataGenerator`. שיטה זו משמשת ליצירת מאגרים של תמונות מעובדות להכשרת מודל למידה עמוקה ממבנה ספרייה.

פירוט הפרמטרים שנשלחו לשיטת "flow_from_directory":

- "train_path" ו- "test_path" (כל אחד בפעולה אחרת) – זהו הנתיב לספרייה המכילה את תמונות האימון.
- "target_size=(150,150)" – מציין את הגודל הרצוי שאליו ישתנו גודל התמונות.
- "batch_size" – זה מציין את מספר התמונות בכל אצווה שתיווצר במהלך האימון. בפרויקט שלי גודל פרמטר זה (בשתי הפעולות) הוא 48, כלומר שיטה זו תיצור אצווה של 48 תמונות כל פעם.
- "class_mode" – פרמטר זה מציין את סוג התוויות שיוחזרו על ידי המחולל. במקרה זה הוא מוגדר להיות 'categorical', משמע המחולל יחזיר עבור כל תמונה תווית מוחלטת. זה מתאים כאשר יש לך מספר קטגוריות ואתה רוצה להכשיר מודל לסיווג רב קטגורי.

הפונקציה flow_from_directory מפשטת את תהליך הטעינה, העיבוד המקדים והגדלת מערכי הנתונים של תמונות להכשרת מודל למידה עמוקה. הוא מספק דרך נוחה ויעילה לעבוד עם נתוני תמונה ותורם לביצועי מודל טובים יותר.

פעולה **compile()** על המודל –

```
# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

הפונקציה compile() ב-keras משמשת להגדרת המודל לאימון. היא מציינת את פונקציית האובדן, האופטימיזציה והמדדים האופציונליים שישמשו במהלך תהליך אימון המודל.

הפרמטרים המועברים לפונקציית compile():

- "loss" – פרמטר זה מגדיר את פונקציית ההפסד שתפעל במהלך האימון. בפרויקט זה השתמשתי ב – "categorical_crossentropy ".
- "optimizer" – האופטימיזציה אשר תפעל ותקבע כיצד המודל יעודכן בהתבסס על ההדרגות המחושבות של פונקציית ההפסד. בפרויקט השתמשתי ב Adam עם קצב למידה של 0.001.
- "metrics" - פרמטר זה מציין את מדדי ההערכה שישמשו במהלך ההדרכה והבדיקות. במקרה זה, הדיוק (accuracy) נבחר כמדד לניטור ביצועי המודל.

על ידי הפעלת compile(), אתה מגדיר את תצורת האימון של המודל, מה שהופך אותו מוכן לקראת האימון.

פעולה **fit()** על המודל –

```
history = model.fit(training_set,
                    validation_data=test_set,
                    epochs=10,
                    steps_per_epoch=len(training_set),
                    validation_steps=len(test_set))
```

הפונקציה fit() ב-keras משמשת לאימון המודל על מערך נתונים נתון. הוא מבצע את האימון כמספר האפוקים שצויינו ומעדכן את הפרמטרים של המודל על סמך נתוני האימון שסופקו.

הפרמטרים המועברים לפונקציה fit():

- "training_set " - זהו מאגר הנתונים אשר משמש לאימון המודל.
- "validation_data" – פרמטר זה מציין את מאגר הנתונים שישמש לאימות במהלך האימון. זה עוזר להעריך את ביצועי המודל על נתונים הנפרדים מ-"training_set ".
- "epochs" – פרמטר זה קובע כמה פעמים כל מאגר הנתונים יעבור במהלך תהליך האימון. במקרה זה הוא מוגדר ל- 10.
- "steps_per_epoch" – מספר השלבים שיש לעבד במהלך כל אפוק. במקרה זה, הוא מוגדר למספר הקבוצות ב-train_set.
- "validation_steps" – מספר השלבים שיש לעבד במהלך האימות. במקרה זה, הוא מוגדר למספר הקבוצות ב-test_set.

במהלך תהליך האימון, משקלי המודל מתעדכנים באמצעות אלגוריתם אופטימיזציה (מוגדר בשיטת compile() של המודל) על סמך ההפסד המחושב והאופטימיזר שצוין. המטרה היא למזער את האובדן ולשפר את ביצועי המודל.

לאחר השלמת האימון, הפונקציה fit() מחזירה אובייקט היסטוריה המכיל מידע על תהליך האימון, כגון ערכי אובדן ודיוק בכל תקופה.

הפונקציה fit() ממלאת תפקיד מרכזי באימון מודלים של למידה עמוקה. הוא משלב עיבוד נתונים, אופטימיזציה, הערכת ביצועים וניטור התקדמות, ומאפשר לך להכשיר מודלים בצורה יעילה.

פונקציית **output()** –

```
# PREDICTION FUNCTION
def output(image_path, class_indices):
    img = load_img(image_path, target_size=(150, 150))
    img_array = img_to_array(img)
    img_array = img_array/255.0
    img_array = np.expand_dims(img_array, axis=0)

    # Use the model to predict the class probabilities of the input image
    class_probabilities = model.predict(img_array)[0]

    # Convert the class probabilities to a class label
    predicted_class_index = np.argmax(class_probabilities)
    predicted_class = class_indices[predicted_class_index]

    print('Predicted class:', predicted_class)
    return predicted_class

# Define the class indices for prediction
class_indices = {0: 'healthy', 1: 'powdery', 2: 'rust'}
```

מטרת הפעולה היא לחזות את תווית הקטגוריה עבור תמונת קלט באמצעות המודל המאומן. הפעולה מקבלת שתי משתנים :

3. `image_path`: נתיב לקובץ של תמונת הקלט שיש לסווג.
4. `class_indices': מילון הממפה בין אינדקסי הקטגוריות לתוויות הקטגוריות.

הפעולה פועלת כך:

9. היא טוענת את התמונה באמצעות הפונקציה `load_img` מהמודול `tensorflow.keras.preprocessing.image` ומשנה את גודלה ל 150*150 פיקסלים.
10. התמונה מומרת למערך NumPy באמצעות הפונקציה `img_to_array` מתוך `tensorflow.keras.preprocessing.image` וערכי הפיקסלים עוברים קנה מידה לטווח [0, 1] על ידי חלוקה ב-255.0.
11. המערך של התמונה מתרחב בציר האינדקס הראשון באמצעות `np.expand_dims`. זה הכרחי כדי להתאים לצורת הקלט שהמודל מצפה לו.
12. פונקציית `predict` של המודל משמשת להשגת הסתברויות המחלקה עבור תמונת הקלט. ההסתברויות החזויות מאוחסנות במשתנה `class_probabilities`.
13. הקטגוריה המנובאת נקבעת על ידי חיפוש האינדקס של הערך המרבי במערך `class_probabilities` באמצעות `np.argmax`.
14. התווית של הקטגוריה המנובאת מתקבלת מהמילון `class_indices' תוך שימוש באינדקס של הקטגוריה המנובאת.
15. התווית של הקטגוריה המנובאת מודפסת למסך.
16. לבסוף התווית של הקטגוריה המנובאת מוחזרת כפלט מהפונקציה.

פעולה זו מאוד חשובה מכיוון שהיא מבצעת את השלב האחרון של התהליך שהוא קבלת תווית הקטגוריה המנובאת.

פעולת **browse_image()** –

```
def browse_image():
    global img
    global canvas
    global prediction_label

    # Open a file dialog to select an image file
filename = filedialog.askopenfilename(initialdir="/",
                                      title="Select Image",
                                      filetypes=(("Image Files",
                                            "*.jpg *.jpeg *.png"),))

    if filename:
        # Open and resize the selected image
        img = Image.open(filename)
        img = img.resize((500, 500))
        img = ImageTk.PhotoImage(img)

        # Update the canvas with the new image
        canvas.image = img
        canvas.create_image(0, 0, anchor="nw", image=img)

        # Make a prediction using the selected image
        prediction_label.config(text="Prediction: " +
                                output(filename, class_indices))
```

פעולה זה מופעלת כאשר המשתמש לוחץ על כפתור "upload" בחלון "prediction_window". כאשר פעולה זו מתחילה לעבוד היא פותחת חלון דו שיח המאפשר למשתמש להעלות תמונה מהמחשב שלו. ברגע שהמשתמש בוחר תמונה הפעולה מאחזרת את נתיב הקובץ של התמונה הנבחרת.
לאחר מכן, הפעולה טוענת את התמונה הנבחרת באמצעות הפונקציה Image.open() מספריית PIL ומשנה את גודלה ל500*500 פיקסלים (הגודל של התמונה אשר תופיעה על המסך).
התמונה עולה על החלון, נשלחת לחיזוי על ידי "output()" ומדפיסה את הקטגוריה המנובאת על פני החלון הגרפי.

הפונקציה הזו חשובה מכיוון שהיא מאפשרת למשתמש לבחור ולטעון קובץ תמונה מהמחשב שלו, זה נותן את האפשרות למשתמש לעבוד עם תמונות שונות ולבדוק את התוצאות של המודל על תמונות חדשות ומגוונות. הפונקציה מציגה את התמונה הנבחרת על החלון "prediction_window". זה מאפשר למשתמש לראות את התמונה שהוא בחר ולוודא שהתמונה תואמת את הציפיות שלו.
הפונקציה קוראת לפעולה output() שמבצעת את סיווג התמונה באמצעות המודל המאומן. זה מאפשר למשתמש לקבל את הקטגוריה המנובאת של התמונה ולראות את התוצאה של המודל על התמונה הנבחרת.

browse_image() מאפשרת למשתמש לבחור, להציג ולבדוק את התמונות שהמודל מסווג. זה מקנה למשתמש חוויה משופרת ומאפשר לו לעבוד בצורה אינטראקטיבית עם המודל והתוצאות שלו.

פעולת open_prediction_window() –

```
def open_prediction_window():
    global prediction_window
    global canvas
    global prediction_label
    global img

    # Close the start window
    start_window.destroy()

    # Create the prediction window
    prediction_window = tk.Tk()
    prediction_window.geometry("750x800")
    prediction_window.title("Plant Disease Recognition - Prediction
                            Window")

    # Label to prompt the user to upload an image
    lblNum = tk.Label(prediction_window, text="Please upload a
                      picture:", height=2, font=("Arial", 24))
    lblNum.pack(side="top")

    # Button to browse and upload an image
    browse_button = tk.Button(prediction_window, text="Upload",
                   command=browse_image, width=10, font=("Arial", 15))
    browse_button.pack(pady=10)

    # Button to reset the image and prediction
    reset_button = tk.Button(prediction_window, text="Reset",
                            command=reset_image, width=10, fg='red',
                            font=("Arial", 15))
    reset_button.pack(pady=10)

    # Canvas to display the uploaded image
    canvas = tk.Canvas(prediction_window, width=500, height=500)
    canvas.pack()

    # Label to display the prediction result
    prediction_label = tk.Label(prediction_window, text="Prediction:"
                                , font=("Arial", 15))
    prediction_label.pack()

    # Button to exit the prediction window
    exit_button = tk.Button(prediction_window, text="Exit",
                           command=exit_prediction_window, width=10,
                           font=("Arial", 15))
    exit_button.place(x=20, y=750)

    # Run the prediction window
    prediction_window.mainloop()
```

פעולה זו נפעלת כאשר כפתור "start" בחלון "start_window" נלחץ. פעולה זו סוגרת את החלון ההתחלתי ופותחת חלון חדש – "prediction_window". הפעולה יוצרת כותרת המזמינה את המשתמש להעלות תמונה על מנת לסווגה ("please upload picture"), מיקום הכותרת הוא למעלה. הפעולה גם יוצרת מקום בו יודפס הקטגוריה שהמודל יחזה, בהתחלה כתוב "Prediction:" וכאשר תעלה תמונה, תודפס הקטגוריה המסווגת.

בנוסף הפעולה יוצרת שלושה כפתורים:

1. כפתור "reset" – כאשר הכפתור נלחץ פעולת reset_image() פועלת:

```
def reset_image():
    canvas.delete("all")
    prediction_label.config(text="Prediction: ")
```

הפעולה מוחקת את התמונה שהמשתמש העלה ואת הקטגוריה המסווגת אשר נכתבה על המסך. כפתור זה מחזיר את החלון למצב המקורי שהיה בהתחלה.

2. כפתור "exit" – כאשר הכפתור נלחץ פעולת "exit_prediction_window()" פועלת:

```
def exit_prediction_window():
    prediction_window.destroy()
```

הפעולה סוגרת את החלון "start_window " כאשר היא פועלת.

3. כפתור "upload" – כאשר כפתור זה נלחץ פעולת "browse_image()" נפעלת.

ספריות

```
from PIL import Image
import random
import matplotlib.pyplot as plt
import os as os
import numpy as np
import tensorflow as tf

from tensorflow.keras.layers import LeakyReLU, Dense, Dropout,
Flatten, Input, BatchNormalization, Activation, Conv2D, MaxPooling2D,
GlobalAveragePooling2D

from tensorflow.keras.models import Model, Sequential, load_model
from tensorflow.keras.utils import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import tkinter as tk
from tkinter import filedialog
from PIL import ImageTk, Image
```

- PIL (Python Imaging Library) – ספרייה המטפלת בתמונות בפיתון. זוהי ספרייה לפתיחה, מניפולציה ושמירה של פורמטים רבים ושונים של קבצי תמונה.
- Random – ספרייה זו משמשת ליצירת מספרים אקראיים וביצוע בחירות אקראיות. בקוד זה, הוא משמש לערבב את שמות הקבצים בספריות ההדרכה והבדיקות.
- matplotlib.pyplot – ספרייה מתמטית ליצירת גרפים מתמטיים.
- Os – ספרייה זו מספקת פונקציות לפעולות קבצים וספריות. בקוד הוא משמש לעבודה עם נתיבי קבצים, רשימת תוכן ספריות ולטפל בשמות קבצים
- Numpy - ספריית קוד מתמטית בפיתון, אשר מספקת תמיכה במערכים גדולים ובמטריצות, וכן מספקת מספר גדול של פעולות מתמטיות שימושיות.
- Tensorflow - היא ספריית למידת מכונה בקוד פתוח שפותחה על ידי Google. הספרייה מספקת תשתית גמישה ויעילה לבניית והדרכה של מודלים של למידה עמוקה. הספרייה תומכת בגרפי חישוב, פעולות למידה עמוקה, פריסת מודלים גמישים להדמיה וניתוח. לספרייה קיים API (ממשק תכנות יישומים) בשם "Keras" לשפות C ו-Python, ועוד רבות אחרות, וזהו הממשק בו אני השתמשתי בפרויקט. ממשקים אלו מפשטים את תהליך הבנייה וההדרכה של רשתות עצביות. מתוך ספרייה זו, בפרויקט שלי אני מייבא פונקציות רבות ביניהן: Dense, Dropout, Flatten, Conv2D, Adam, ImageDataGenerator, load_img ועוד.
- Tkinter - ספריית Python סטנדרטית ליצירת ממשקי משתמש גרפיים (GUI). היא מספקת קבוצה של כלים ווידג'טים המאפשרים לבנות יישומים אינטראקטיביים עם לחצנים, תוויות, תיבות טקסט ורכיבי GUI אחרים.

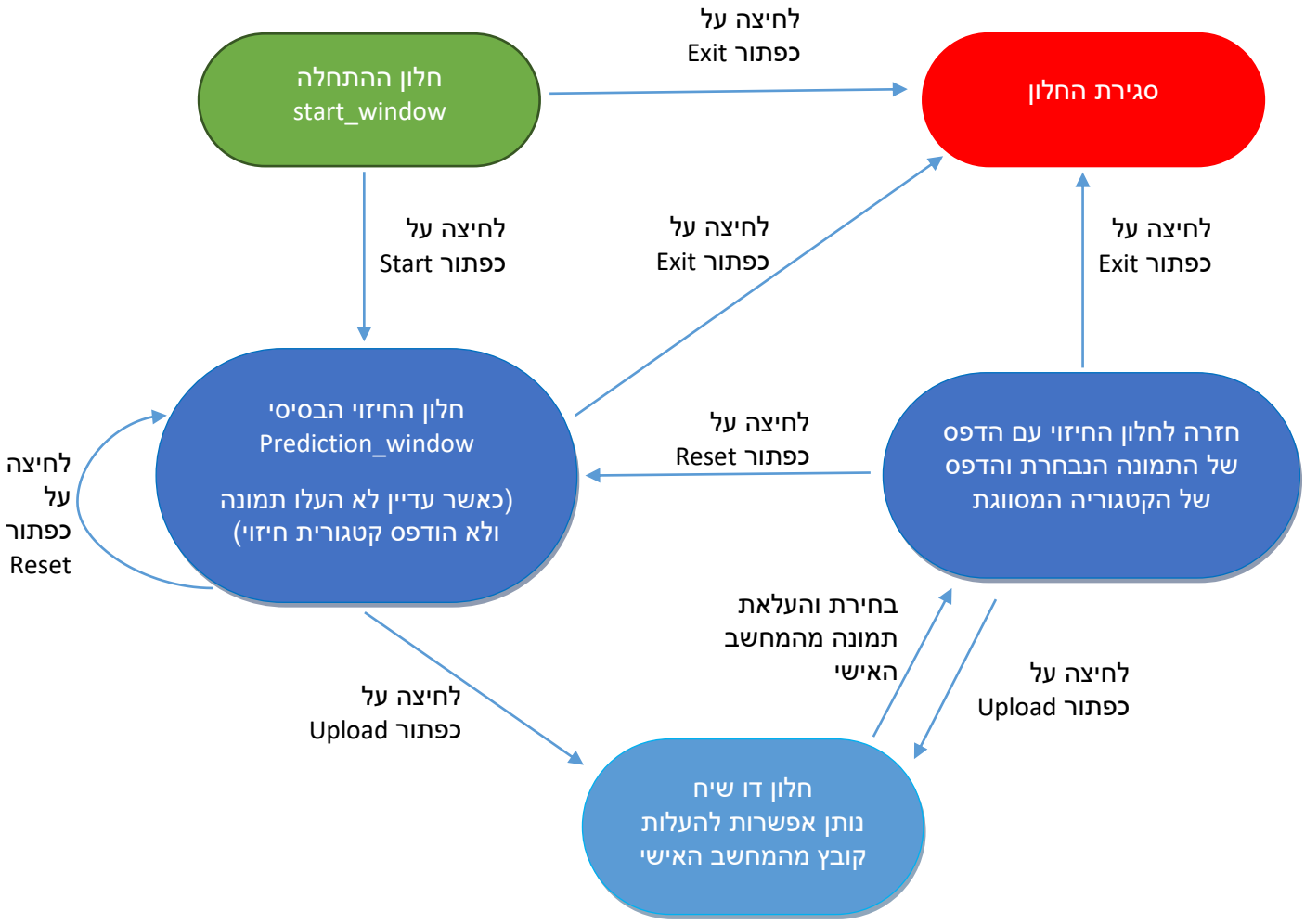
מדריך למשתמש:

הוראות התקנה

על מנת להתחיל ולסווג תמונות יש להוריד את סביבת העבודה PyCharm – לחץ כאן להורדה.

לאחר שהורדת את סביבת העבודה יש לפתוח את הפרויקט מהתיקייה, ללחוץ על כפתור ההרצה ואפשר להתחיל להעלות תמונות ולסווג.

תרשים המסכים



מסכי הפרויקט

מסך ראשון – חלון התחלה (start_window).

חלון ההתחלה משמש כממשק המשתמש הראשוני עבור היישום. תפקידו לספק ממשק פשוט עם הודעת פתיחה וכפתור "התחל" כדי להתחיל את תהליך החיזוי.

חלון ההתחלה מאפשר למשתמש לשלוט מתי להתחיל בתהליך החיזוי על ידי לחיצה על כפתור "התחל". הוא מספק הפרדה ברורה בין הממשק הראשוני לחלון החיזוי, מה שהופך את האפליקציה ליותר ידידותית.

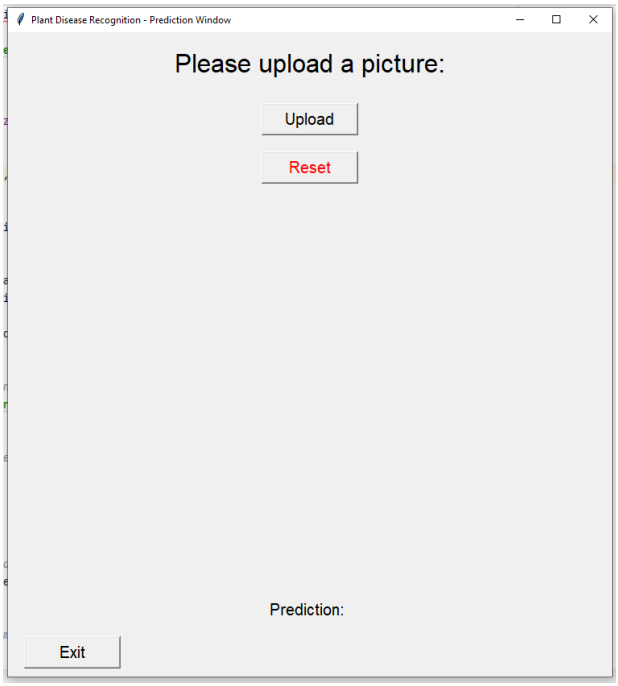
מסך שני – חלון החיזוי (prediction_window).

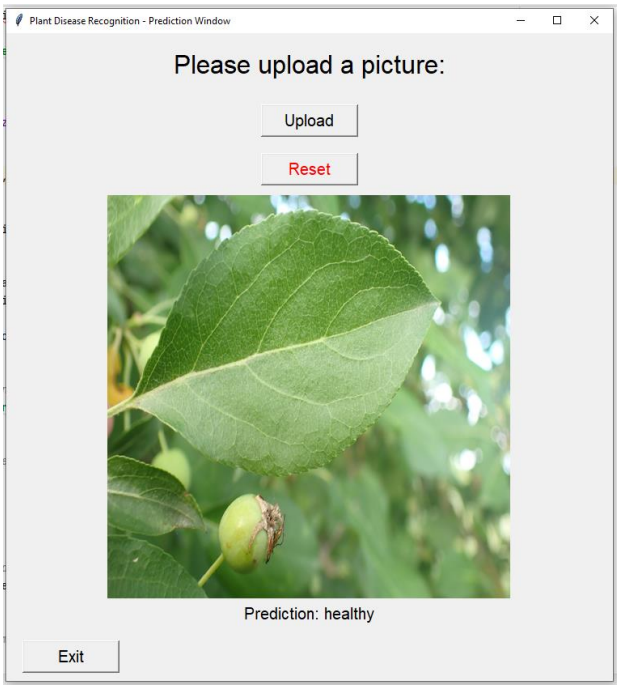
חלון החיזוי הוא ממשק המשתמש הראשי ביישום שבו משתמשים יכולים להיכנס ולחזות תמונות. תפקידו להציג את התמונה שנבחרה על ידי המשתמש, לספק את התוצאה החזויה לתמונה ולאפשר למשתמשים לבצע פעולות כמו העלאת תמונות חדשות, איפוס התצוגה ויציאה מהאפליקציה.

חלון החיזוי משרת את המטרות הבאות:

- משתמשים יכולים ללחוץ על כפתור "העלה" כדי לדפדף ולבחור קובץ תמונה מהמערכת המקומית שלהם. לאחר מכן, התמונה שנבחרה מוצגת בחלון.
- לאחר העלאת תמונה, האפליקציה מבצעת חיזוי על התמונה וסיווגה יודפס על החלון.
- משתמשים יכולים ללחוץ על כפתור "איפוס" כדי לנקות את הקנבס ולהסיר את התמונה ותווית החיזוי המוצגת, מה שמאפשר להם להעלות תמונה חדשה ולבצע חיזוי חדש.
- משתמשים יכולים ללחוץ על כפתור "יציאה" כדי לסגור את חלון החיזוי ולסיים את הסיווג.

על ידי מתן פונקציונליות אלה, חלון החיזוי מאפשר למשתמשים ליצור אינטראקציה עם האפליקציה, לחקור תמונות שונות ולקבל תחזיות לזיהוי מחלות צמחים.



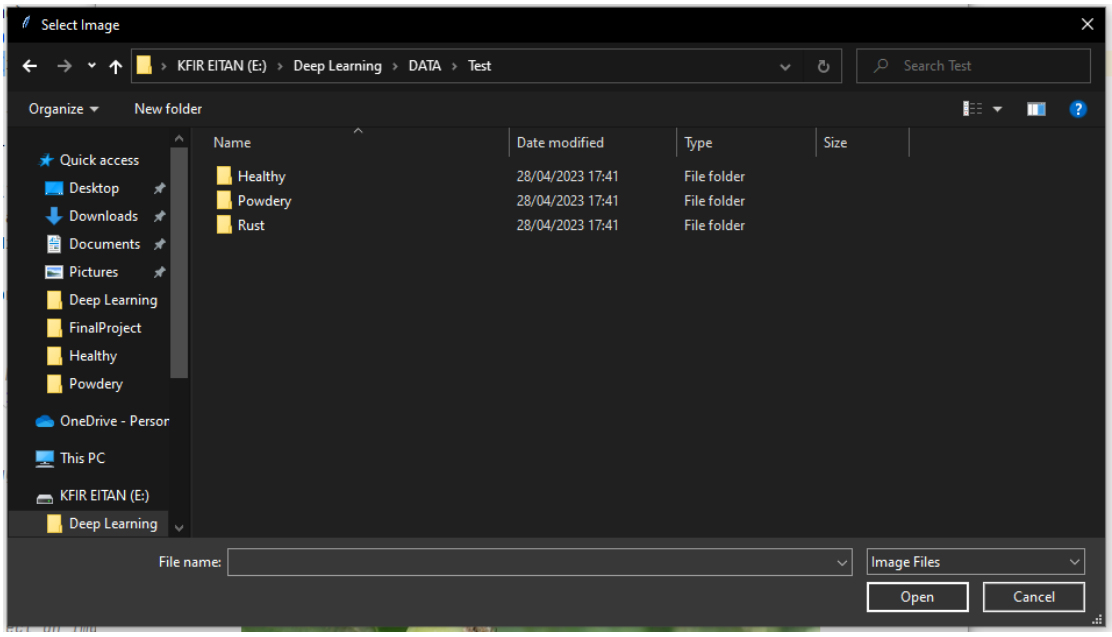


חלון דו שיח –

תיבת הדו-שיח של הקובץ ממלאת תפקיד מכריע ביישום זיהוי מחלות צמחים. הוא משמש כדי לאפשר למשתמשים לגלוש ולבחור קבצי תמונה מהמערכת המקומית שלהם.

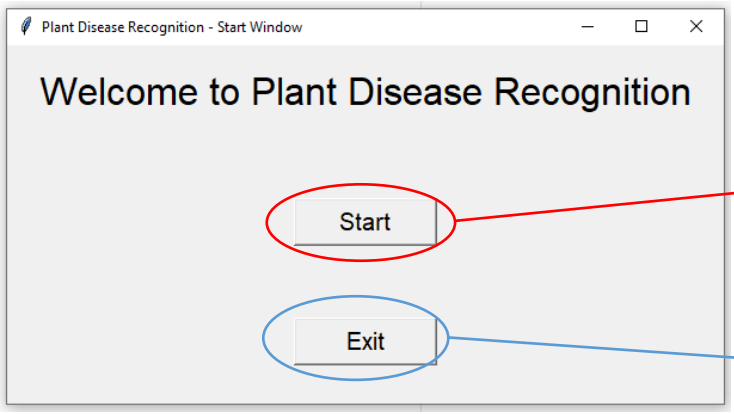
ברגע שהמשתמש לוחץ על כפתור upload חלון הדו שיח נפתח ומשם יש לו את האפשרות לנווט במערכת הקבצים שלו ולבחור קובץ תמונה.

תיבת הדו-שיח של הקבצים מבטיחה דרך ידידותית ואינטואיטיבית למשתמשים ליצור אינטראקציה עם האפליקציה, ומאפשרת להם לבחור את התמונה הספציפית שהם רוצים לנתח מבלי לדרוש קלט ידני של נתיב הקובץ.



הסברי כפתורים ולחצנים בחלונות השונים

חלון ההתחלה, start_window –



כפתור Start, בעת לחיצתו חלון ההתחלה ייסגר וחלון הסיווג, prediction_window, יפתח.

כפתור Exit, בעת לחיצתו חלון ההתחלה ייסגר ולא ימשיך לחלון הבא.

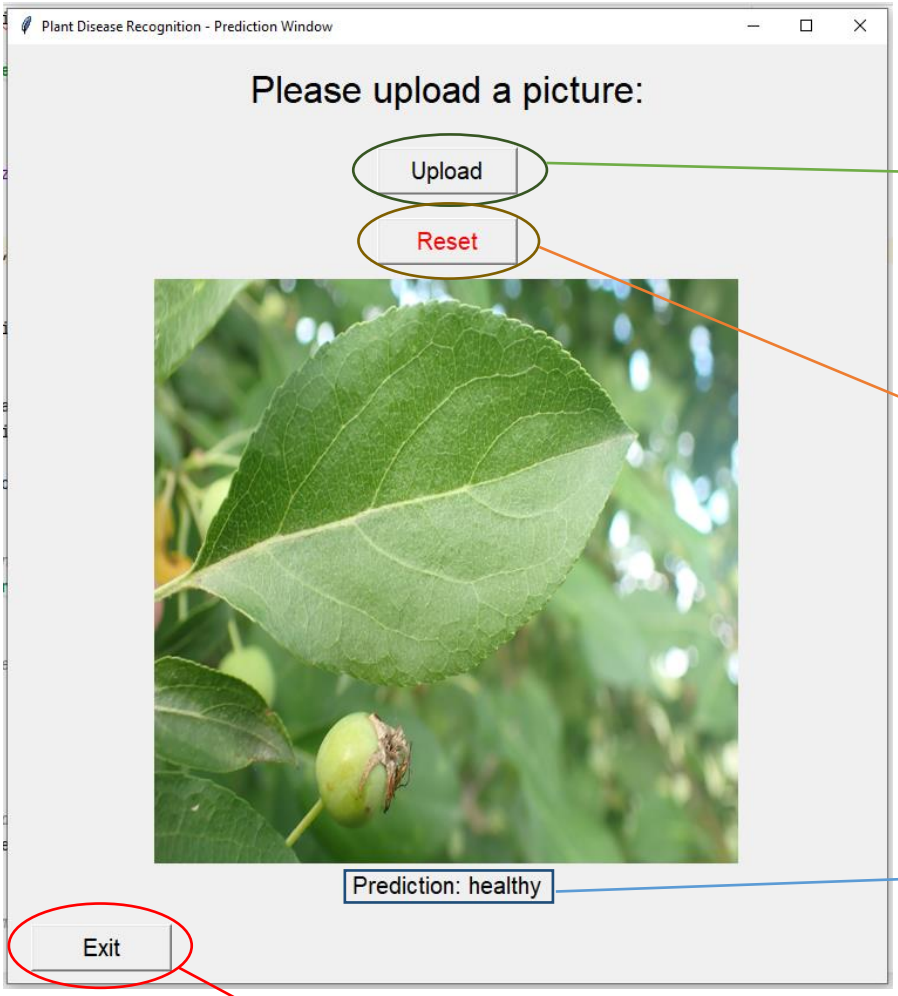
חלון החיזוי, prediction_window, לפני שהועלתה תמונה לסיווג –

כפתור upload, בעת לחיצתו תיפתח תיבת דו שיח בה המשתמש יכול להעלות קובץ תמונה מהמכשיר הפרטי שלו.

במצב זה כפתור הreset לא יעשה דבר מכיוון שלא הועלתה תמונה לסיווג.

כפתור Exit, בעת לחיצתו חלון החיזוי ייסגר.

חלון החיזוי, prediction_window, לאחר שהועלתה תמונה לסיווג –



כפתור upload, בעת לחיצתו תיפתח תיבת דו שיח בה המשתמש יכול להעלות קובץ תמונה מהמכשיר הפרטי שלו.

כפתור reset, בעת לחיצתו התמונה שהמשתמש העלה ומופיעה על המסך תעלם ביחד עם סיווגה.

בתיבה זו תודפס הקטגוריה שהמודל סיווג על התמונה שהמשתמש העלה.
[healthy, powdery, rust]

כפתור Exit, בעת לחיצתו חלון החיזוי ייסגר.

רפלקציה:

העבודה על הפרויקט במסגרת שיעורי לימוד מכונה הייתה מאוד מעניינת ומאתגרת. במהלך הפרויקט נדרשתי לערוך מחקר מעמיק בתחום זיהוי מחלות אצל צמחים ובפתרונות טכנולוגיים המתאימים למשימה. בעזרת השיעורים בבית הספר ולמידה עצמית בבית, הצלחתי ליצור מודל אשר מצליח במשימתו ומסווג את מצב הצמח על ידי צילום העלה שלו.

בעקבות יישום הפרויקט, נחשפתי לעולם מחלות הצמחים שהיה חדש עבורי צברתי ידע כללי חדש. במהלך הלימוד עבור פיתוח המודל נחשפתי לשפת פייתון אשר לא הכרתי לפני, נחשפתי לכל התהליך של אימון המודל ועל הדרך שבה הוא מצליח לסווג את התמונות.
בנוסף, קיבלתי כלים חדשים אשר בעתיד יכולים לעזור לי כמו למידה עצמאית בבית, מחקר ולמידה עצמאי וחיפוש אחר פתרונות לבעיות שעלו במהלך יישום הפרויקט.
למדתי על מגוון עצום של טכנולוגיות: החל ממכונות לומדות וראייה ממוחשבת ועד ידע תיאורטי כמו גיאומטריה חישובית. מעבר לכלים הטכנולוגיים, רכשתי יכולות כגון משמעת עבודה, ניהול פרויקטים, הצבת יעדים ועמידה בזמנים.

אני אקח איתי לחיים את היכולת לחקור וללמוד באופן עצמאי נושא שכלל לא הכרתי לפני, ניהול הזמן על מנת להגיש את הפרויקט בזמן שהוקצב לנו וכמובן את היכולת לפתח בשפת פייתון מודל לימוד מכונה.

לאורך העבודה על הפרויקט עלו כמה קשיים ואתגרים שנאלצתי להתמודד איתם. בהתחלה היה קושי לבחור את נושא הפרויקט. רציתי שהוא יהיה נושא אשר מעניין אותי ובנוסף שיהיה גם אקטואלי לחיינו. בנוסף, כאשר הורדתי את מאגר הנתונים נקלעתי לקושי עם גודלי התמונות במאגר. גודלי התמונות היו מאוד גדולים וזה השפיע על זמן לימוד המודל היכולת הסיווג שלו. בעקבות קושי זה חיפשתי דרך בה אפשר להתעלות עליה וכתבתי קוד אשר מקטין את כל גודלי המונות לגודל אחיד וקטן משמעותית ממה שהיה לפני כן.

לו הייתי מתחיל את הפרויקט היום הייתי מתחיל לעבוד עליו מוקדם יותר ממה שהתחלתי בפועל, ייתכן שבתחילת העבודה הייתי מעט שאנן לגביה וככל שהתעמקתי בעבודה הבנתי שהיה לי כדאי להתחיל מוקדם יותר ולתכנן את הזמנים בצורה יעילה יותר.

לסיכום, העבודה על הפרויקט הייתה מאוד מהנה ומלמדת. היא לימדה אותי דברים חדשים גם בהיבט הטכנולוגי ובהיבט האישי. בהיבט הטכנולוגי למדתי המון על שפת הפיתון אשר זוהי פעם הראשונה שתכנתי בה, למדתי לעבוד עם סביבת העבודה PyCharm ולמדתי ידע תיאורטי אשר אותו מימשתי בבניית המודל. בהיבט האישי הפרויקט הקנה לי משמעת עבודה, הצבת יעדים ועמידה בהם ואיך לנהל את הזמן שלי באופן אפקטיבי יותר. למדתי המון מעבודת הגמר ואני מקווה שבעתיד יהיה לי ההזדמנות לעשות עוד פרויקט.

ביבליוגרפיה:

1. A Comprehensive Guide on Optimizers in Deep Learning
https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/#Conclusion

2. How Activation Functions Work in Deep Learning
https://www.kdnuggets.com/2022/06/activation-functions-work-deep-learning.html

3. What is the Softmax Function?
https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

4. A Basic Introduction to Activation Function in Deep Learning
https://www.analyticsvidhya.com/blog/2022/03/a-basic-introduction-to-activation-/function-in-deep-learning

5. Understanding Loss Function in Deep Learning
https://www.analyticsvidhya.com/blog/2022/06/understanding-loss-function-in-/deep-learning

6. Descending into ML: Training and Loss
https://developers.google.com/machine-learning/crash-course/descending-into-ml/training-and-loss

7. Dropout in Neural Networks
https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9

8. Regulation
https://www.merriam-webster.com/dictionary/regulation

9. Image Augmentation on the fly using Keras ImageDataGenerator!
https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-/using-keras-imagedatagenerator

10. בשימוש שלבי יצירת חלון TkInter GUI
http://blog.csit.org.il/UpLoad/FilesUpload/%D7%A9%D7%9C%D7%91%D7%99%D7%99%D7%A6%D7%99%D7%A8%D7%AA%D7%97%D7%9C%D7%95%D7%9F-%D7%93%D7%A3%D7%9C%D7%AA%D7%9C%D7%9E%D7%99%D7%93GUI%D7%91%D7%A9%D7%99%D7%9E%D7%95%D7%A9TkInter-Python.pdf